Avoiding recurrence in multiscale environments

E. Francomano, A. Tortorici, E. Toscano

Rapporto Tecnico N.:
RT-ICAR-PA-06-08 **novembre 2006**

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Avoiding recurrence in multiscale environments

E. Francomano[1,2], A. Tortorici[1,2], E. Toscano[1,2]

Rapporto Tecnico N.:
RT-ICAR-PA-06-08 **novembre 2006**

[1] Istituto di Calcolo e Reti ad Alte Prestazioni, ICAR-CNR, Sezione di Palermo Viale delle Scienze edificio 11 90128 Palermo
[2] Università degli Studi di Palermo Dipartimento di Ingegneria Informatica Viale delle Scienze, edificio 6, 90128 Palermo

Abstract. In designing multiscale models, process informations are reported at multiple levels of resolution commonly involving operations such as discrete convolution and upsampling. In this paper a computational tool following the evolution of a process across the scales by using suitable vectors weight and involving only initial data sampling is provided. A novel non recursive algorithm generating the vectors weight is proposed and computational efficiency is reached when more sets of initial data sampling are considered. *B*-spline functions are well known in performing representations at different scales; experiments involving cardinal *B*-spline functions are provided validating the non recursive scheme interesting mono and multidimensional data sampling.

Keywords: multiscale methods; graphical display algorithm; quasi-interpolant operator; wavelet discrete transform.

1. Introduction

The general idea to represent a process at multiple scales is well known. In the last decades multiscale methods have been used in numerous areas of applied mathematics as signals analysis, statistics, image processing and numerical analysis. Multiscale methods are based on data approximation of a given problem at various resolution levels by travelling through scales. Efficient scaling mechanisms are required to concentrate on some details or to get a better overview of the phenomena and the convolution and the upsampling are operations commonly employed in multiresolution schemes.

In this paper an efficient computational strategy is proposed to avoid the intrinsic recurrence across the scales so noticeably reducing the computational effort when more sets of initial data sampling are considered. Namely, a computational law is carried out generating vectors weight, which enable to directly pass to a desired level of resolution, and by involving only the initial data sampled. The algorithm is formulated in multivariate settings. The recurrence-free computational scheme is adopted in the interpolatory graphical display scheme [1, 3], quasi-interpolant approximations [1, 3] and wavelet algorithms [1, 3, 11], processes all strongly characterized by convolution and upsampling operations. A pre-processing stage must be performed to generate a tree of vectors weight regarding the prefixed resolution level: in the graphical display algorithm and in approximating scattered data across the scales by means of a quasi-interpolant operator a binary vectors weight tree is generated, whilst a quad-tree has to be built to produce the vectors weight in the recurrence-free wavelet algorithms.

A general framework of scale-space representation is in the context of *B*-spline functions [1, 4, 10]. Some vectors weight and function values are provided working with cardinal *B*-spline functions. The applications previously discussed are performed in univariate and bivariate *B*-spline spaces [2]. The paper is organized as follows. In section 2 the non recursive multiscale algorithm is described and formulated in multidimensional spaces. Section 3 regards study cases. Subsection 3.1 is devoted to the interpolatory graphical display algorithm and three levels of vectors weight binary tree, generated by means of cardinal *B*-spline function of order $m = 4$, are reported. Moreover, some function values at different resolution levels are carried out. In subsection 3.2 the non recursive algorithm is examined in the approximation at different scales of scattered data by means of a quasi-interpolant operator based on centered cardinal *B*-spline function of order $m = 4$. Subsection 3.3 provides the recurrence-free scheme for wavelet reconstruction algorithm. The cardinal *B*-spline function of order $m = 3$ and the corresponding compactly supported semi-orthogonal wavelet function are used in generating two levels of the quad-tree of vectors weight.

2. A non recursive multiscale algorithm

A hierarchical representation of a phenomenon can be constructed by means of *dilations* and *translations* in the bases $\{\phi(2^j x - k)\}$, $j, k \in \mathrm{Z}$, governed by the two-scale relation at level *j*:

$$\phi(2^j x) = \sum_k p_k \phi(2^{j+1} x - k). \tag{1}$$

A function $\phi(x) \in L^2(\mathrm{R})$ satisfying (1) is called *scaling function* and $p = \{p_k\}$ is the corresponding coefficients sequence [1, 3, 11]. In the following *p* will be considered with finite size s_p. A possible representation of a phenomenon $f(x)$ can be expressed at level *j* as follows:

$$f^{(j)}(x) = \sum_k c_k^{(j)} \phi(2^j x - k). \tag{2}$$

That is, a phenomenon on a coarse grid $2^{-j_1} k$ (or low resolution) can be represented on a fine grid $2^{-j_2} k$ (or high resolution) with $j_1 < j_2$. By fixing two successive levels involving coefficients $c^{(j)} = \{c_k^{(j)}\}$ and $c^{(j+1)} = \{c_k^{(j+1)}\}$ the following relation holds [3]:

$$c_k^{(j+1)} = \sum_l p_{k-2l} c_l^{(j)}. \tag{3}$$

The computation of $c_k^{(j+1)}$ requires only an *upsampling* followed by a *discrete convolution*. The upsampling is achieved by treating the index *l* in $c_l^{(j)}$ as an even index and inserting a zero between $c_l^{(j)}$ and $c_{l+1}^{(j)}$ for each *l*, that is:

$$\{\tilde{c}_l^{(j)}\} := \begin{cases} c_k^{(j)} & \text{if} \quad l = 2k \\ 0 & \text{if} \quad l = 2k+1. \end{cases} \tag{4}$$

Therefore, by taking into account the sequence $\{\tilde{c}_l^{(j)}\}$ the formula (3) involves only a discrete convolution $c_k^{(j+1)} = \sum_l p_{k-l} \tilde{c}_l^{(j)}$. In figure 1 the two operations are displayed.

$$\begin{tikzcd} \boxed{c^{(j)}} \arrow[r] & \textcircled{\uparrow 2} \arrow[r] & \boxed{*p} \arrow[r] & \boxed{c^{(j+1)}} \end{tikzcd}$$

FIG. 1. Computation of $c^{(j+1)}$ by means of *upsampling*, $(\uparrow 2)$, followed by *discrete convolution*, $(*)$.

The coefficients $\left\{c_k^{(j+1)}\right\}$ are recursively computed by involving the upsampled values of the previous level. In the following, a recurrence-free variant of the process is proposed generating coefficients by means of suitable vectors weight operating only on a subset of $\left\{c_k^{(j_0)}\right\}$ at the initial level j_0. Without lose of generality in the following $j_0 = 0$.

In order to compute each $c_k^{(j+1)}$, with $j+1 > j_0$, a vector weight $v_t^{(j+1)}$ is involved where the index t is a sequence of binary digits related to the index k and the resolution level. Namely, at level *j*, 2^j vectors $v_t^{(j)}$ have been generated and 2^j matrices $G_t^{(j)}$ must be performed as:

$$G_t^{(j)} = (g_t^{(j)}(i_1, i_2)) := v_t^{(j)} \otimes p, \qquad i_1 = 1,\dots,s_t^{(j)}, i_2 = 1,\dots,s_p, \tag{5}$$

where $s_t^{(j)}$ is the size of the vector $v_t^{(j)}$, s_p is the size of *p*.

Hence, a new vector is generated:

$$w_t(r) = \sum_k g_t^{(j)}(k, r-k+1), \qquad r = 1,\dots, s_t^{(j)} + s_p - 1, \tag{6}$$

by assuming equal to zero the addends $g_t^{(j)}(k, r-k+1)$ with $r-k+1 \le 0$.

The even-numbered components of w_t give rise to the vector weight $v_{\tau_0}^{(j+1)}$ whilst the odd ones are in the vector $v_{\tau_1}^{(j+1)}$ where τ_0 and τ_1 are $j+1$ binary digits obtained juxtaposing 0 and 1 ahead of *t*, respectively.

At the beginning, a vector $v^{(0)}$, without *t* index, composed by the values of the scaling function ϕ computed in prefixed knots must be taken into account.

Ergo, the coefficient $c_k^{(j+1)}$ can be generated as follows:

$$c_k^{(j+1)} = \sum_{l=s_t^{(j+1)}}^{1} v_t^{(j+1)}(s_t^{(j+1)} - l + 1) c_{q-l}^{(j_0)}, \quad q = \left\lceil \frac{k}{2^{j+1}} \right\rceil. \tag{7}$$

The index *t* of the vector $v_t^{(j+1)}$ is the binary value of $k \, mod \, 2^{j+1}$ by involving $j+1$ binary digits.

In figure 2 the vectors weight are schematized as nodes of a binary tree.

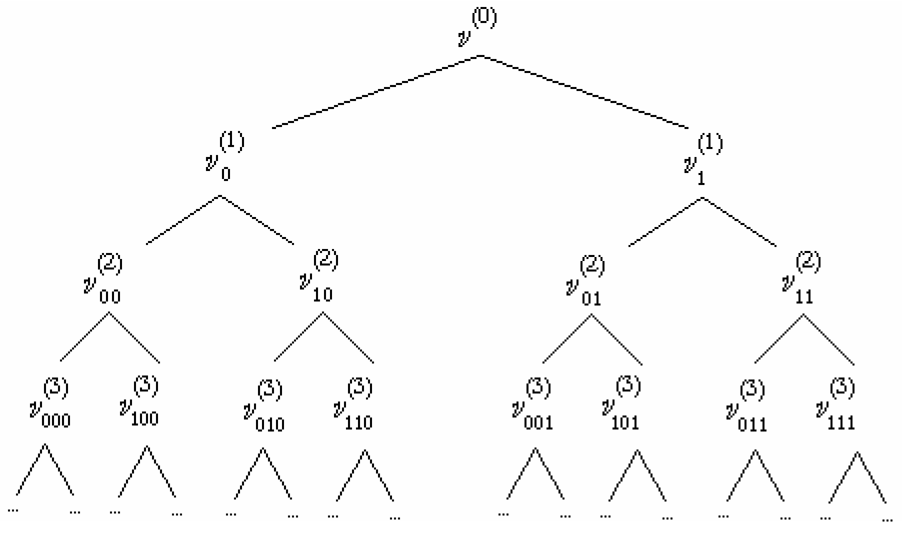


FIG. 2. Binary tree of vectors weight.

At level $j+1$, an amount of $\sum_{i=1}^{2^j} s^{(j)}_{t_i} s_p$ products and $\sum_{i=1}^{2^j} (s^{(j)}_{t_i} + s_p - 1)$ adds is required; moreover, $s^{(j+1)}_{t} - 1$ adds and $s^{(j+1)}_{t}$ products are involved in (7). The advantage of the proposed computational tool is in performing the overall computations only once and using them for each initial data set.

Dealing with functions of *d*-variables tensor products are necessary and the algorithm can be reformulated as:

$$c^{(j+1)}_{k_1,k_2,\dots,k_d} = \sum_{l_1=s^{(j+1)}_{t_1}}^{1} \sum_{l_2=s^{(j+1)}_{t_2}}^{1} \dots \sum_{l_d=s^{(j+1)}_{t_d}}^{1} (v^{(j+1)}_{t_1} \otimes v^{(j+1)}_{t_2} \otimes \dots \otimes v^{(j+1)}_{t_d})(\xi_1,\xi_2,\dots,\xi_d)\, c^{(j_0)}_{q_1-l_1,q_2-l_2,\dots,q_d-l_d} \text{ ,(8)}$$

where $q_i = \left\lceil \frac{k_i}{2^{j+1}} \right\rceil$, t_i is the binary value of $k_i \, mod \; 2^{j+1}$ and $\xi_i = s^{(j+1)}_{t_i} - l_i + 1$, $i = 1,\dots,d$.

The binary tree of vectors weight is taken into account in generating the coefficients in *d*-space involving, this time, matrices whose size depends on the employed vectors in (8).

3. Study cases

3.1 Interpolatory graphical display algorithm

This section is devoted to apply the just reported computational tool to the interpolatory graphical display algorithm [1]: by fixing as initial data set $c^{(j_0)} = \left\{ f^{(j_0)}_{k} \right\} = \left\{ f\left(\frac{k}{2^{j_0}} \right) \right\}$ the aim is to compute the values $f^{(j_1)} = \left\{f_k^{(j_1)}\right\} = \left\{f\left(\frac{k}{2^{j_1}}\right)\right\}$, $\forall j_1 > j_0$, and for values of j sufficiently large $f^{(j)}$ is adequate to approximate the function $f(x)$.

The process is described by choosing as scaling function the cardinal *B*-spline function of order *m*, $N_m(x)$, for which the two scale relation holds [1, 3, 4, 10]:

$$N_m(2^j x) = \sum_{h=0}^{m} 2^{-m+1}\binom{m}{h} N_m(2^{j+1}x - h) = \sum_{h=0}^{m} p_{m,h} N_m(2^{j+1}x - h) \tag{9}$$

and

$$f_k^{(j+1)} = \sum_l c_l^{(j+1)} N_m(k-l), \tag{10}$$

where

$$c_l^{(j+1)} = \sum_h 2^{-m+1}\binom{m}{l-h}\tilde{c}_h^{(j)} = \sum_h p_{m,l-h}\tilde{c}_h^{(j)}. \tag{11}$$

3.1.1 The recurrence-free scheme

In the following the non recursive interpolatory graphical display algorithm is outlined.

A pre-processing stage must be performed to generate the vectors weight of a prefixed resolution level.

ALGORITHM

Input: *m*, $v^{(0)}(i) = N_m(i) \quad i = 1,\ldots,m-1$, $\left\{f_k^{(0)}\right\}_{k=0}^{n}$, final level j_f

Output: $\left\{f_k^{(j_f)}\right\}_{k=0}^{n_f}$

Vectors weight pre-computation:

```
1. Computation of $\{p_{m,h}\}_{h=0}^{m}$
2. For each level $j = 0,\ldots,j_f - 1$:
    2.1. For each $i = 0,\ldots,2^j - 1$:
        2.1.1. $t$ is the binary value of $i$ by involving $j$ binary digits
        2.1.2. $G_t^{(j)} = (g_t^{(j)}(i_1,i_2)) = v_t^{(j)} \otimes \{p_{m,h}\}_{h=0}^{m}$, $i_1 = 1,\ldots,s_t^{(j)}, i_2 = 1,\ldots,m+1$
```

2.1.3. $w_t(r) = \sum_{k=1}^{s_t^{(j)}} g_t^{(j)}(k, r-k+1), \quad r = 1,\dots, s_t^{(j)} + m$

2.1.4. $v_{\tau_0}^{(j+1)}(r) = w_t(2r), \quad r = 1,\dots, \left\lfloor \frac{s_t^{(j)} + m}{2} \right\rfloor$

2.1.5. $v_{\tau_1}^{(j+1)}(r) = w_t(2r-1), \quad r = 1,\dots, \left\lfloor \frac{s_t^{(j)} + m + 1}{2} \right\rfloor$

end of pre-computation

For each data set $c^{(j_0)} = \left\{ f\left(\frac{k}{2^{j_0}}\right) \right\}_{k=0}^{n}$:

3. $f_k^{(j_f)} = \sum_{l=s_t^{(j_f)}}^{1} v_t^{(j_f)}(s_t^{(j_f)} - l + 1) f_{q-l}^{(j_0)}, \quad k = 0,\dots,n_f, \; q = \left\lceil \frac{k}{2^{j_f}} \right\rceil$

When the vectors weight are performed only at most m products are required to compute each value $f_k^{(j)}$ and the computational complexity is independent across the scale. Therefore, appreciable reductions in the computation occur when more sets of initial data are considered.

By considering the cardinal B-spline function of order $m = 4$ [1, 4, 10], the initial vector $v^{(0)}$ is composed by the B-spline function values in the integer knots in $]0,4[$, $v^{(0)} = \left(\frac{1}{6}, \frac{4}{6}, \frac{1}{6}\right)$ and $p_{4,0} = \frac{1}{8}$, $p_{4,1} = \frac{4}{8}$, $p_{4,2} = \frac{6}{8}$, $p_{4,3} = \frac{4}{8}$, $p_{4,4} = \frac{1}{8}$. Hence:

$$G^{(0)} = v^{(0)} \otimes \{p_{4,h}\}_{h=0}^{4} = \frac{1}{6 \cdot 8} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

and

$$w = \frac{1}{6 \cdot 8}(1,8,23,32,23,8,1).$$

Consequently, the two vectors $v_0^{(1)}$ and $v_1^{(1)}$ are generated by extracting the entries of the vector w located in the even and odd positions, respectively.

At the next levels $j > 0$, the following operations must be performed:

$$G_t^{(j)} = v_t^{(j)} \otimes \{p_{4,h}\}_{h=0}^{4}$$

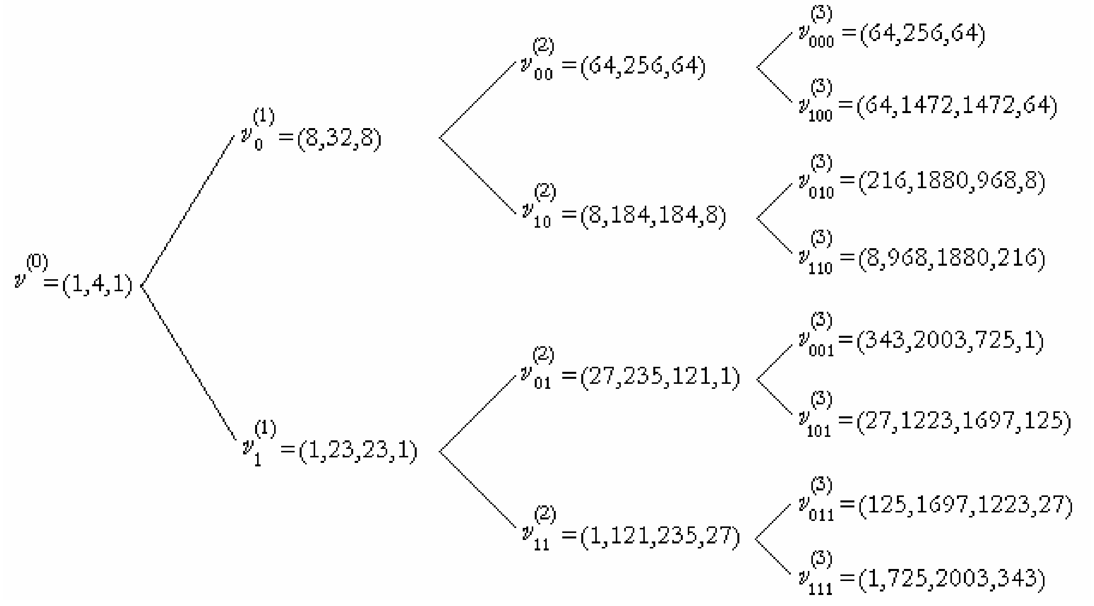
and

$$w_t(r) = \sum_{k=1}^{s_t^{(j)}} g_t^{(j)}(k, r-k+1), \qquad r = 1,\dots,s_t^{(j)}+4\,,$$

where $s_t^{(j)} = 4$ or $s_t^{(j)} = 3$.

Now, the entries of the vector w_t give rise to the vectors $v_{\tau_0}^{(j+1)}$ and $v_{\tau_1}^{(j+1)}$.

In figure 3 the vectors weight regarding the cardinal *B*-spline function of order $m = 4$ are reported for $j = 0,1,2,3$ and for the sake of simplicity the common denominator 6×8^j is dropped in the vectors weight.



$v^{(0)}=(1,4,1)$
$v_0^{(1)}=(8,32,8)$
$v_1^{(1)}=(1,23,23,1)$
$v_{00}^{(2)}=(64,256,64)$
$v_{10}^{(2)}=(8,184,184,8)$
$v_{01}^{(2)}=(27,235,121,1)$
$v_{11}^{(2)}=(1,121,235,27)$
$v_{000}^{(3)}=(64,256,64)$
$v_{100}^{(3)}=(64,1472,1472,64)$
$v_{010}^{(3)}=(216,1880,968,8)$
$v_{110}^{(3)}=(8,968,1880,216)$
$v_{001}^{(3)}=(343,2003,725,1)$
$v_{101}^{(3)}=(27,1223,1697,125)$
$v_{011}^{(3)}=(125,1697,1223,27)$
$v_{111}^{(3)}=(1,725,2003,343)$

FIG. 3. Three levels of the vector weights tree generated by cardinal *B*-spline function of order *m*=4.

In figures 4 and 5 the initial function values and the vectors weight involved in the computation of some function values $f_k^{(j)}$ are shown.

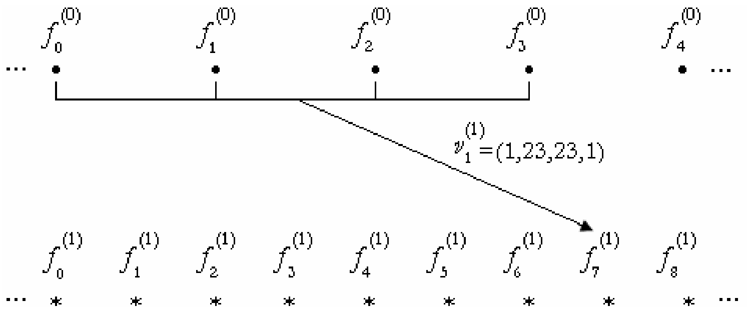


FIG. 4. Generation of $f_7^{(1)}$.

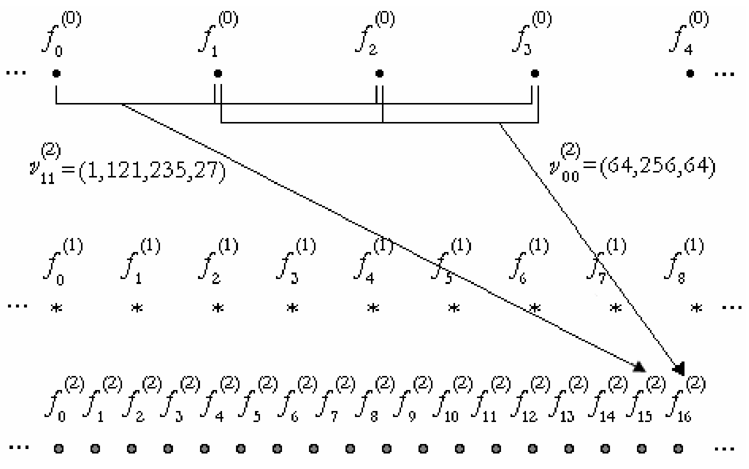

$f_0^{(0)}$ $f_1^{(0)}$ $f_2^{(0)}$ $f_3^{(0)}$ $f_4^{(0)}$
$v_{11}^{(2)}=(1,121,235,27)$
$v_{00}^{(2)}=(64,256,64)$
$f_0^{(1)}$ $f_1^{(1)}$ $f_2^{(1)}$ $f_3^{(1)}$ $f_4^{(1)}$ $f_5^{(1)}$ $f_6^{(1)}$ $f_7^{(1)}$ $f_8^{(1)}$
$f_0^{(2)}$ $f_1^{(2)}$ $f_2^{(2)}$ $f_3^{(2)}$ $f_4^{(2)}$ $f_5^{(2)}$ $f_6^{(2)}$ $f_7^{(2)}$ $f_8^{(2)}$ $f_9^{(2)}$ $f_{10}^{(2)}$ $f_{11}^{(2)}$ $f_{12}^{(2)}$ $f_{13}^{(2)}$ $f_{14}^{(2)}$ $f_{15}^{(2)}$ $f_{16}^{(2)}$

FIG. 5. Generation of $f_{15}^{(2)}$ and $f_{16}^{(2)}$.

In the following the bivariate formulation is reported:

$$f_{k_1,k_2}^{(j+1)} = \sum_{l_1=s_{t_1}^{(j+1)}}^{1} \sum_{l_2=s_{t_2}^{(j+1)}}^{1} (v_{t_1}^{(j+1)} \otimes v_{t_2}^{(j+1)})(\xi_1,\xi_2) f_{q_1-l_1,q_2-l_2}^{(j_0)}, \tag{12}$$

where $q_i = \left\lceil \frac{k_i}{2^{j+1}} \right\rceil$, t_i is the binary value of $k_i \, mod \; 2^{j+1}$ and $\xi_i = s_{t_i}^{(j+1)} - l_i + 1$, $i = 1,2$.

By choosing $m = 4$ the following two-dimensional vectors weight are generated at level $j = 1$:

$$v_0^{(1)} \otimes v_0^{(1)} = \frac{1}{6^2 \times 8^2} \begin{pmatrix} 64 & 256 & 64 \\ 256 & 1024 & 256 \\ 64 & 256 & 64 \end{pmatrix},$$

$$v_0^{(1)} \otimes v_1^{(1)} = (v_1^{(1)} \otimes v_0^{(1)})^T = \frac{1}{6^2 \times 8^2} \begin{pmatrix} 8 & 184 & 184 & 8 \\ 32 & 736 & 736 & 32 \\ 8 & 184 & 184 & 8 \end{pmatrix},$$

$$v_1^{(1)} \otimes v_1^{(1)} = \frac{1}{6^2 \times 8^2} \begin{pmatrix} 1 & 23 & 23 & 1 \\ 23 & 529 & 529 & 23 \\ 23 & 529 & 529 & 23 \\ 1 & 23 & 23 & 1 \end{pmatrix}$$

and the formula (12) is applied in computing the function value $f_{7,9}^{(1)}$:

$$f_{7,9}^{(1)} = \sum_{l_1=4}^{1}\sum_{l_2=4}^{1}(v_1^{(1)} \otimes v_1^{(1)})(5-l_1,5-l_2)f_{4-l_1,5-l_2}^{(0)} = \frac{1}{6^2 \times 8^2}[(f_{0,1}+f_{0,4}+f_{3,1}+f_{3,4})+$$
$$+23(f_{0,2}+f_{0,3}+f_{1,1}+f_{1,4}+f_{2,1}+f_{2,4}+f_{3,2}+f_{3,3})+529(f_{1,2}+f_{1,3}+f_{2,2}+f_{2,3})].$$

3.2 Data approximation by quasi-interpolant

When scattered data are considered the approximation results can be appreciable improved by applying a *quasi-interpolant operator* Qf [1, 5, 6, 7]. The main advantage is that this operator is local, i. e. the value of $(Qf)(x)$ depends only on values of f in a neighbourhood of x. Moreover, Qf has a rather small infinity norm, so that it is nearly optimal approximant [1, 8, 9].

By considering the centered cardinal B-spline function of order m a quasi-interpolant operator can be expressed as:

$$(Qf)(x) = \sum_{l \in Z} \lambda_l(f) N_m(x + 2^{-1}m - l), \tag{13}$$

where $\lambda_l(f)$ is a local linear functional.

In the following the quasi-interpolant operator generated by the functional [1]:

$$\lambda_l(f) = \frac{1}{6}[-f_{l-1} + 8f_l - f_{l+1}], \quad l \in \mathbb{Z}, \tag{14}$$

and the centered cardinal B-spline function of order $m = 4$ is considered:

$$(Qf)(x) = \sum_{l \in Z} \frac{1}{6}\left[-f_{l-1} + 8f_l - f_{l+1}\right] N_4(x + 2 - l). \tag{15}$$

The non recursive algorithm previously exposed can be applied in approximating a function with the Qf operator (15). Namely, the non recursive algorithm generates the matrices:

$$H_t^{(j)} = (h_t^{(j)}(i_1,i_2)) := v_t^{(j)} \otimes v_{\lambda_l}, \qquad i_1 = 1,\dots,s_t^{(j)}, i_2 = 1,2,3, \tag{16}$$

where $v_t^{(j)}$ are the vectors of the pre-computed binary tree in subsection 3.1.1, $v_{\lambda_l} = (-1/6, 4/3, -1/6)$ and:

$$\overline{w}_t\,(r) = \sum_k h_t^{(j)}(k, r-k+1), \qquad r = 1, \ldots, s_t^{(j)} + 2, \tag{17}$$

by assuming equal to zero the addends $h_t^{(j)}(k, r-k+1)$ with $r-k+1 \le 0$.

By extracting the even and odd-indexed entries of $\overline{w}_t$ the vectors $\overline{v}_{\tau_0}^{(j+1)}$ and $\overline{v}_{\tau_1}^{(j+1)}$ are generated and involved in the computation of the data sampling at the level $j+1$:

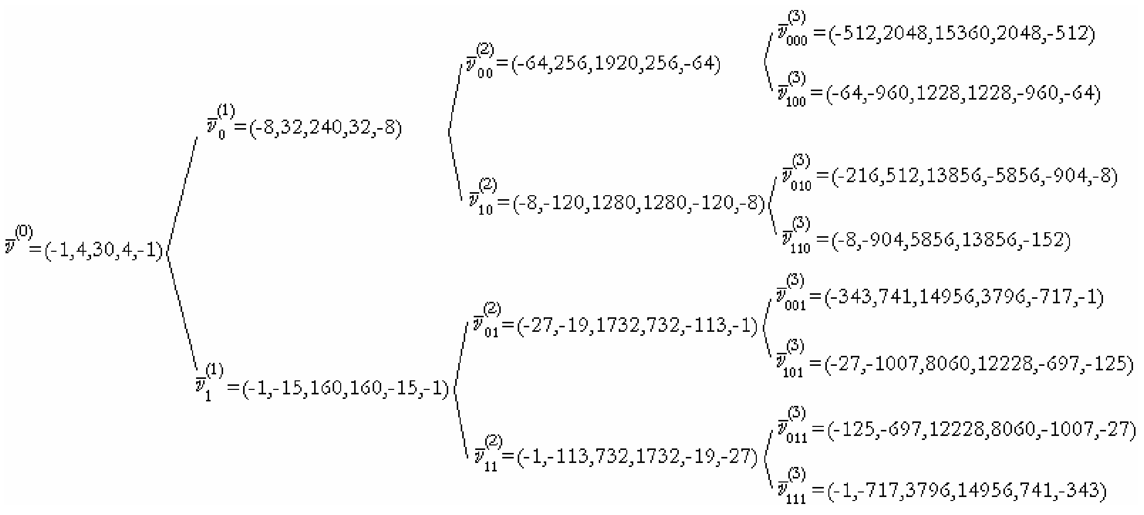
$$f_k^{(j+1)} = \sum_{l=s_t^{(j+1)}}^{1} \overline{v}_t^{(j+1)}(s_t^{(j+1)} - l + 1) f_{q-l+1}^{(0)}, \qquad q = \left\lceil \frac{k}{2^{j+1}} \right\rceil + 2, \tag{18}$$

where $s_t^{(j+1)}$ is the size of the vector $\overline{v}_t^{(j+1)}$.
In figure 6 the binary tree of the vectors weight is reported for $j = 0,1,2,3$ and for the sake of simplicity the common denominator $6^2 \times 8^j$ is dropped.
In the following the $f_7^{(1)}$ and $f_{15}^{(2)}$ values are reported by adopting the formula (18):

$$f_7^{(1)} = \frac{1}{6^2 \times 8}(-f_1 - 15f_2 + 160f_3 + 160f_4 - 15f_5 - f_6),$$

$$f_{15}^{(2)} = \frac{1}{6^2 \times 8^2}(-f_1 - 113f_2 + 732f_3 + 1732f_4 - 19f_5 - 27f_6).$$

Note that the centered cardinal *B*-spline functions involve initial data values symmetrically placed around the computed value.



$\overline{v}^{(0)}$=(-1,4,30,4,-1)
$\overline{v}_0^{(1)}$=(-8,32,240,32,-8)
$\overline{v}_1^{(1)}$=(-1,-15,160,160,-15,-1)
$\overline{v}_{00}^{(2)}$=(-64,256,1920,256,-64)
$\overline{v}_{10}^{(2)}$=(-8,-120,1280,1280,-120,-8)
$\overline{v}_{01}^{(2)}$=(-27,-19,1732,732,-113,-1)
$\overline{v}_{11}^{(2)}$=(-1,-113,732,1732,-19,-27)
$\overline{v}_{000}^{(3)}$=(-512,2048,15360,2048,-512)
$\overline{v}_{100}^{(3)}$=(-64,-960,1228,1228,-960,-64)
$\overline{v}_{010}^{(3)}$=(-216,512,13856,-5856,-904,-8)
$\overline{v}_{110}^{(3)}$=(-8,-904,5856,13856,-152)
$\overline{v}_{001}^{(3)}$=(-343,741,14956,3796,-717,-1)
$\overline{v}_{101}^{(3)}$=(-27,-1007,8060,12228,-697,-125)
$\overline{v}_{011}^{(3)}$=(-125,-697,12228,8060,-1007,-27)
$\overline{v}_{111}^{(3)}$=(-1,-717,3796,14956,741,-343)

FIG. 6. The binary tree generated by using the quasi-interpolant operator of formula (15).

3.3 Wavelet decomposition and reconstruction algorithms

When a given problem is investigated in a wavelet multiscale framework, as well known [1, 3, 11] the problem modeling function $f^{(j+1)}(x) \in V_{j+1}$ is decomposed as:

$$f^{(j+1)}(x) = f^{(j)}(x) + g^{(j)}(x), \tag{19}$$

where $f^{(j)}(x) \in V_j$, $g^{(j)}(x) \in W_j$ and:

$$\begin{cases} f^{(j)}(x) = \sum_k c_k^{(j)} \phi(2^j x - k) \\ g^{(j)}(x) = \sum_k d_k^{(j)} \psi(2^j x - k). \end{cases} \tag{20}$$

Hence, $c^{(j)} = \{c_k^{(j)}\}$ and $d^{(j)} = \{d_k^{(j)}\}$ are used instead of $f^{(j)}(x)$ and $g^{(j)}(x)$ as output and input data sequences in the decomposition and reconstruction algorithms, respectively.

Namely, to find $f^{(j)}(x)$ and $g^{(j)}(x)$ from $f^{(j+1)}(x)$ the following recursive relations among coefficients hold:

$$\begin{cases} c_k^{(j)} = \sum_l a_{l-2k} c_l^{(j+1)} \\ d_k^{(j)} = \sum_l b_{l-2k} c_l^{(j+1)}. \end{cases} \tag{21}$$

Note that both $c^{(j)}$ and $d^{(j)}$ are obtained from $\{c_l^{(j+1)}\}$ by a discrete convolution with filters sequences $a = \{a_{-k}\}$ and $b = \{b_{-k}\}$, respectively, followed by a *downsampling* (usually denoted by $\downarrow 2$) achieved by discarding the odd-numbered components.

In order to recover $f^{(j+1)}(x)$ from $f^{(j)}(x)$ and $g^{(j)}(x)$ the input sequences $c^{(j)}$ and $d^{(j)}$ are firstly upsampled and subsequently convolved with the scaling and wavelet filters sequences, $p = \{p_k\}$ and $q = \{q_k\}$, respectively:

$$c_k^{(j+1)} = \sum_l (p_{k-2l} c_l^{(j)} + q_{k-2l} d_l^{(j)}). \tag{22}$$

In figure 7 decomposition and reconstruction algorithms are schematically displayed.

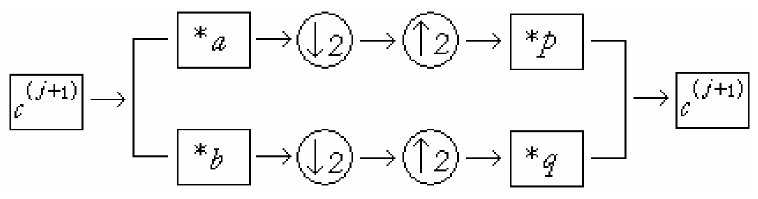


FIG. 7. Wavelet decomposition-reconstruction scheme.

3.3.1 The recurrence-free scheme

The recursion can be avoided by employing the computational tool described in section 2 and in the following the wavelet reconstruction algorithm is formulated. Namely, at each level *j*, the matrices $G_t^{(j)(S)}$ and $G_t^{(j)(W)}$ involving the *scaling* and *wavelet sequences* must be generated:

$$G_t^{(j)(S)} = v_t^{(j)(S)} \otimes p\,, \tag{23}$$

$$G_t^{(j)(W)} = v_t^{(j)(S)} \otimes q \tag{24}$$

and

$$w_t^{(S)}(r) = \sum_k g_t^{(j)(S)}(k, r-k+1)\,, \tag{25}$$

$$w_t^{(W)}(r) = \sum_k g_t^{(j)(W)}(k, r-k+1). \tag{26}$$

The even-numbered components of the vectors $w_t^{(S)}$ and $w_t^{(W)}$ give rise to the vectors weight $v_{\tau_0}^{(j+1)(S)}$ and $v_{\tau_0}^{(j+1)(W)}$, whilst the remaining compose the vectors $v_{\tau_1}^{(j+1)(S)}$ and $v_{\tau_1}^{(j+1)(W)}$.

At the beginning $v^{(0)(S)}$ is a vector whose entries are the values of the scaling function ϕ computed in prefixed knots. The vectors weight pre-processing step is schematically described by means of a quad-tree (figure 8).

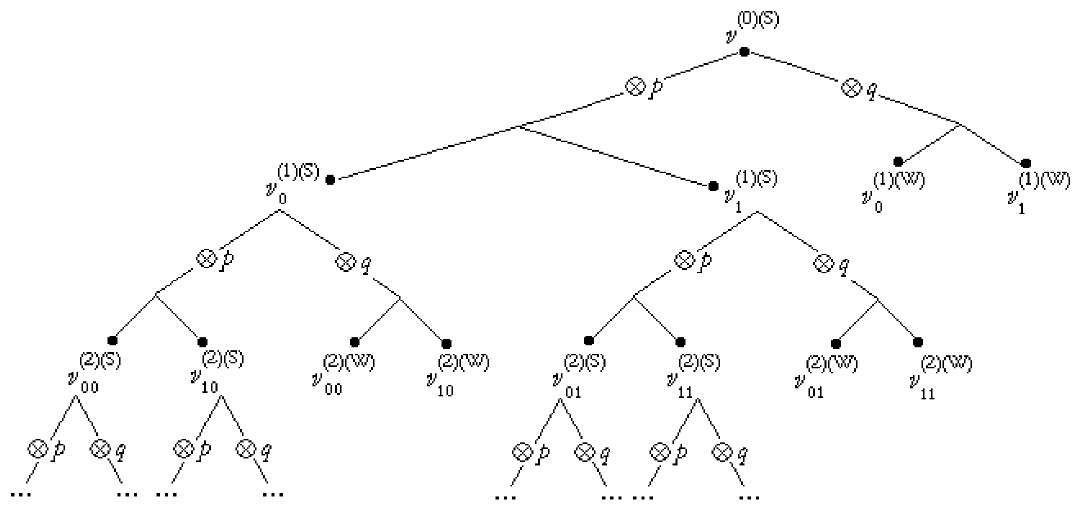


FIG. 8. Quad-tree of vectors weight in wavelet reconstruction algorithm.

The non recursive algorithm just described is now investigated by considering the multiresolution analysis in which $\{V_j\}_{j\in Z}$ are the cardinal *B*-spline functions spaces.

In the following the compactly supported semi-orthogonal wavelet $\psi_m(x)$, corresponding to the scaling function $N_m(x)$, are taken into account. This function, with support $[0, 2m-1]$, is unique and is given by [1]:

$$\psi_m(x) = \sum_{n=0}^{3m-2} q_n N_m(2x-n), \tag{27}$$

where

$$q_n = \frac{(-1)^n}{2^{m-1}} \sum_{l=0}^{m} \binom{m}{l} N_{2m}(n+1-l), \qquad n = 0,\dots,3m-2. \tag{28}$$

By referring to $N_3(x)$, the scaling sequence is: $p_{3,0} = \frac{1}{4}$, $p_{3,1} = \frac{3}{4}$, $p_{3,2} = \frac{3}{4}$, $p_{3,3} = \frac{1}{4}$; the corresponding wavelet function is $\psi_3(x)$ and wavelet sequence is: $q_{3,0} = \frac{1}{4}$, $q_{3,1} = -\frac{29}{4}$, $q_{3,2} = \frac{147}{4}$, $q_{3,3} = -\frac{303}{4}$, $q_{3,4} = \frac{303}{4}$, $q_{3,5} = -\frac{147}{4}$, $q_{3,6} = \frac{29}{4}$, $q_{3,7} = -\frac{1}{4}$.

At the beginning $v^{(0)(S)} = (N_3(1), N_3(2)) = \left(\frac{1}{2}, \frac{1}{2}\right)$. In figure 9 two levels of the vectors weight quad-tree is reported and for the sake of simplicity the common denominators, 2×4^j and $2\times 4^j \times 5!$ for scaling and wavelet vectors weight, respectively, are dropped.

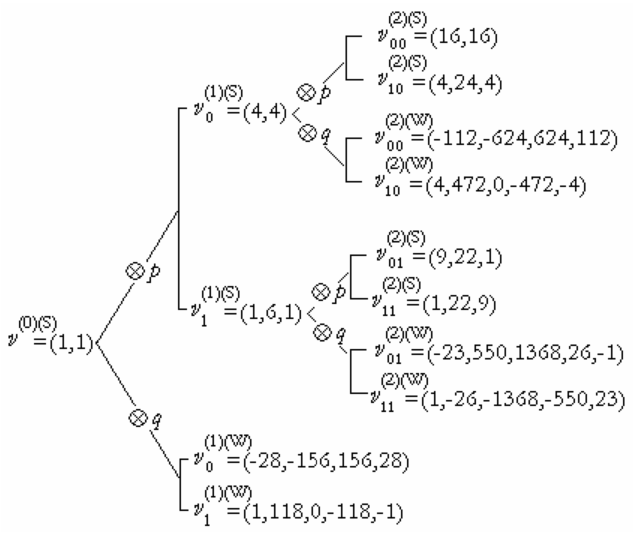

$v^{(0)(S)} = (1,1)$
$\otimes p$
$\otimes q$
$v_0^{(1)(S)} = (4,4)$
$v_1^{(1)(S)} = (1,6,1)$
$v_0^{(1)(W)} = (-28,-156,156,28)$
$v_1^{(1)(W)} = (1,118,0,-118,-1)$
$v_{00}^{(2)(S)} = (16,16)$
$v_{10}^{(2)(S)} = (4,24,4)$
$v_{00}^{(2)(W)} = (-112,-624,624,112)$
$v_{10}^{(2)(W)} = (4,472,0,-472,-4)$
$v_{01}^{(2)(S)} = (9,22,1)$
$v_{11}^{(2)(S)} = (1,22,9)$
$v_{01}^{(2)(W)} = (-23,550,1368,26,-1)$
$v_{11}^{(2)(W)} = (1,-26,-1368,-550,23)$

FIG. 9. Two levels of the quad-tree generated by cardinal *B*-spline function of order *m*=3.

4. Conclusions

In this paper an efficient computational strategy is proposed to avoid the intrinsic recurrence across the scales when multiscale processes are investigated. The main result of the proposed tool is in formulating a computational law which generates vectors weight, enabling to directly pass to a desired level of resolution, and involves only the initial data sampled. The computational law is used in the interpolatory graphical display algorithm, in approximating scattered data across the scales by means of a quasi-interpolant operator and in wavelet reconstruction algorithm. Vectors weight and function values are provided working with cardinal *B*-spline functions. The algorithm is also formulated in multivariate settings.

REFERENCES

[1] C. K. CHUI, *An introduction to wavelets*, Academic Press, Boston, 1992.

[2] C. K. CHUI, *Multivariate splines*, SIAM, Philadelphia, 1988.

[3] C. K. CHUI, *Wavelets: a mathematical tool for signal analysis*, SIAM, Philadelphia, 1997.

[4] C. DE BOOR, *A pratical guide to splines*, Springer-Verlag, Berlin, 1978.

[5] C. DE BOOR, *The quasi-interpolant as a tool in elementary polynomial spline theory*, Approximation Theory, G. G. Lorentz, ed., Academic Press, New York, 1973.

[6] C. DE BOOR, *Quasiinterpolants and approximation power of multivariate splines*, Computation of Curves and Surfaces, W. Dahmen, M. Gasca and C. A. Micchelli (eds.), Kluwer, 1990.

[7] C. DE BOOR AND G. FIX, *Spline approximation by quasiinterpolants*, Journal of Approximation Theory, 8 (1973), pp. 19-45.

[8] P. SABLONNIÈRE, *Quadratic spline quasi-interpolants on bounded domains of* $\mathbb{R}^d$*, d=1,2,3*, Rendiconti del Seminario Matematico Università e Politecnico Torino, 61 (2003), pp. 229-246.

[9] P. SABLONNIÈRE, *Univariate spline quasi-interpolants and applications to numerical analysis*, Rendiconti del Seminario Matematico Università e Politecnico Torino, 61 (2003), pp. 211-222.

[10] L. SCHUMAKER, *Spline Functions: Basic Theory*, Wiley-Interscience, New York, 1981.

[11] G. STRANG AND T. NGUYEN, *Wavelets and Filter Banks*, Wellesley, Cambridge, 1996.