ANIMATION OF FRACTAL OBJECTS

Jean-François COLONNA

GSV-LACTAMME
Ecole Polytechnique
91128 Palaiseau Cédex
France

Tel : 33 1 60 19 40 53
Fax : 33 1 69 41 33 92

Abstract : This paper describes a new method for the generation of fractal objects like mountains and clouds. It is based on the superposition of n-dimensional meshes. It is shown that with meshes of dimension higher than 2, it allows the animation of fractal objects, and for example the simulation of cloud dynamics and earthquakes.

Résumé : Ce papier décrit une nouvelle méthode de génération d'objets fractals, tels des montagnes ou des nuages. Elle repose sur la superposition de maillages n-dimensionnels. Il est montré que, pour des dimensions supérieures à 2, elle permet l'animation de ces objets, et par exemple la simulation de la dynamique des nuages ou encore celle des tremblements de terre.

Key words : animation, clouds, cloud dynamics, earthquakes, fractal objects, mountains.

1-INTRODUCTION :

1.1-The recursive subdivision algorithm : to generate, for example, mountains surfaces using fractal techniques, they are well known methods like Fourier transform of white noise and recursive subdivision [1][2]. The first one is slow, and the second one shows artifacts. Let's recall it. This algorithm starts with a triangle or a quadrilateral : for this last case, let $V_1V_2V_3V_4$ be the initial two-dimensional quadrilateral ; on each side V_iV_{i+1} let's choose (deterministically or randomly) a point $P_{i\,i+1}$, and inside of $V_1V_2V_3V_4$ a fifth point P. Then the five points P, P_{12}, P_{23}, P_{34} and P_{41} are randomly moved along the third dimension, thus giving birth to four new "three-dimensional" quadrilaterals $V_1P_{12}PP_{41}V_1$, $P_{12}V_2P_{23}PP_{12}$, $P_{23}V_3P_{34}PP_{23}$ and $P_{34}V_4P_{41}PP_{34}$ (see the figure 1). Then the process is iterated recursively for each new quadrilateral ; at the end, we obtain an irregular surface. This process frequently shows artefacts (like creases of slope discontinuities [2]) and cannot model easily smooth terrains.

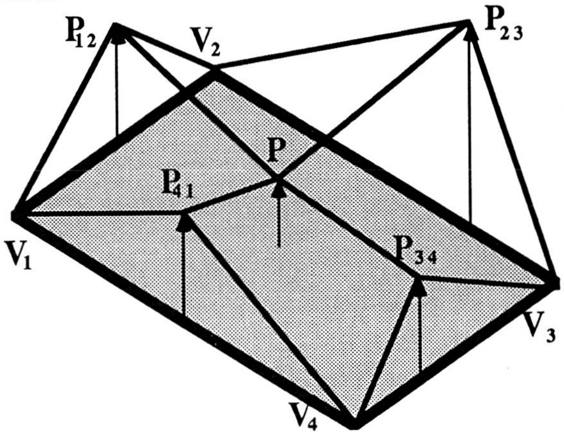


Figure 1 : the subdivision algorithm.

1.2-The superposition of two-dimensional meshes : We propose a new method based on the **superposition of n-dimensional meshes**. To be clear, let's describe it with a two-dimensional example. We shall use a set of square meshes $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m\}$ of decreasing sizes (the smallest one is on the order of the pixel size). We never require coincidences between nodes belonging to various meshes, unlike the recursive subdivision algorithm ; thus the computations of all the meshes will be independent. For the mesh $\mathcal{M}_k$ ($k \in [1,m]$) we generate on each vertex V_{ij} a random value RDN(i,j,k,S) as a function of i and j (the coordinates relating to the mesh $\mathcal{M}_k$), k (the rank of the mesh) and S (a seed for the random generator). Then the mesh is "rasterized" in the following way : if the current point P(x,y) is a vertex V_{ij}, it will receive the value RDN(i,j,k,S), otherwise, its value will be interpolated between the values of its neighbours. Finally, a two-dimensional field is obtained by superposing and adding point by point all the rasterized meshes ; the figure 2 shows the three first steps and the last one of this iterative process. Then this field can be visualized as a two-dimensional field, as a three-dimensional surface (the value at the point P(x,y) giving the third coordinate), or again transformed and combined with other two-dimensional fields (the figure 3 shows an interpolation between such a fractal field and a geometric shape ; the first appendix gives a view of the available tools for the manipulation of two-dimensional fields).

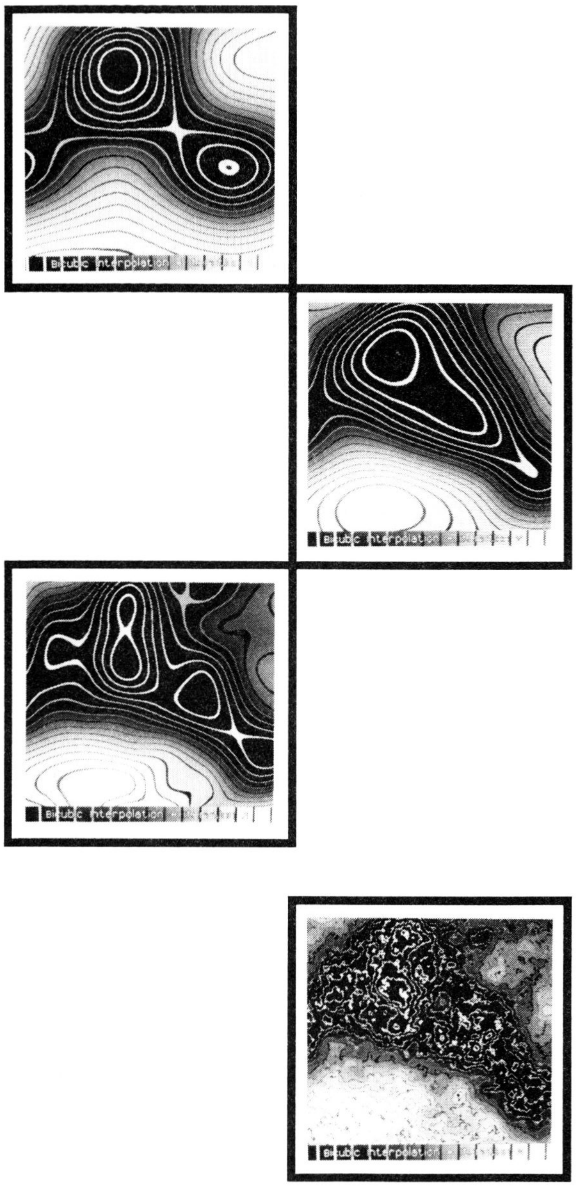


Figure 2 : the three first steps ($\mathcal{M}_1$, $\mathcal{M}_1+\mathcal{M}_2$, $\mathcal{M}_1+\mathcal{M}_2+\mathcal{M}_3$) and the last one ($\mathcal{M}_1+\mathcal{M}_2+...+\mathcal{M}_N$ with N=12) of the iterative process with a bi-cubic interpolation (512x512, 256 colors).

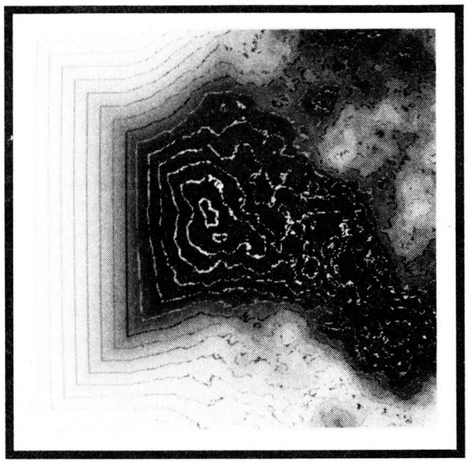


Figure 3 : an interpolation between a two-dimensional fractal field (obtained by the superposition method) and a geometrical shape (512x512, 256 colors).

With this two-dimensional example, it may be seen that this method gives us four "degrees of freedom" :

-the **geometry of the meshes**,
-the **interpolation function**,
-the **random generator**,
-and finally the **mapping function** between meshes and space coordinates.

Two interpolation functions have been implemented. The first one is a **bi-linear** one (in this two-dimensional example), and the second one is a **bi-cubic** one (see the second appendix) . When used with meshes such that :

$$\text{size}(\mathcal{M}_k) = \text{size}(\mathcal{M}_{k-1})/2$$

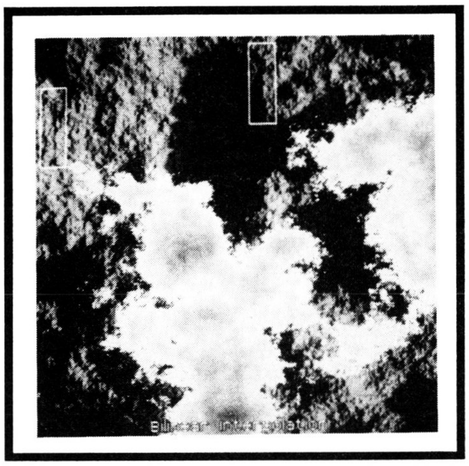
and such that one vertex of $\mathcal{M}_k$ coincide with one vertex of $\mathcal{M}_{k-1}$, the bi-linear interpolation can simulate the subdivision algorithm. The figure 4 shows the same three-dimensional scene computed with both interpolation functions. The results of the bi-linear one shows clearly slope discontinuities (two of them are enhanced inside white rectangles), when the bi-cubic one looks more realistic.

2-THE SUPERPOSITION OF n-DIMENSIONAL MESHES :

2.1-Definitions : let $\mathcal{E}$ be a subset of $\mathbf{R^n}$ with coordinates $(x_1,x_2,...,x_n)$. Let $\{\mathcal{M}_1,\mathcal{M}_2,...,\mathcal{M}_k,...,\mathcal{M}_m\}$ be a family of **hypercellular meshes** of decreasing sizes :

$$\text{size}(\mathcal{M}_1) > \text{size}(\mathcal{M}_2) > ... > \text{size}(\mathcal{M}_k) > ... > \text{size}(\mathcal{M}_m).$$

It is important to notice that this family is not obtained with a recursive subdivision and that all its elements are independent. Let **RDN($x_1,x_2,...,x_n$,k,S)** be a random generator ; its arguments are the coordinates **($x_1,x_2,...,x_n$)**, the rank **k** of the current mesh $\mathcal{M}_k$ and a seed **S**.



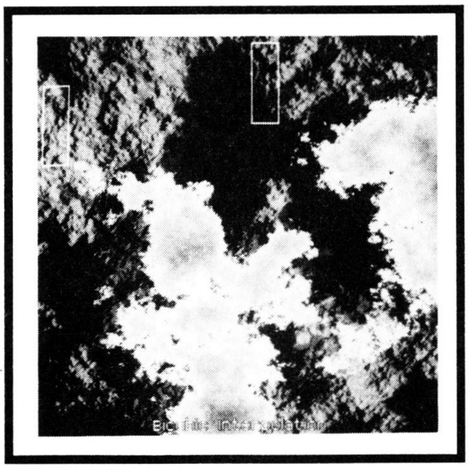


Figure 4 : comparison between the bi-linear and the bi-cubic interpolations with the help of two upper views of a fractal terrain ; the bi-linear interpolation clearly shows, inside the white rectangles, slope discontinuities enhanced by the effect of a light source located at the right side (512x512, 256 colors).

2.2-The hyperfield : for each point ($x_1,x_2,...,x_n$) of $\mathcal{E}$ and for each mesh $\mathcal{M}_k$ we define a scalar hyperfield f_k :

$f_k(x_1,x_2,...,x_n,k,S)$=**RDN($x_1,x_2,...,x_n$,k,S)** if the point ($x_1,x_2,...,x_n$) of $\mathcal{E}$ is a vertex of the current mesh $\mathcal{M}_k$,
=**Interpolation** between the values **RDN($x_1,x_2,...,x_n$,k,S)** computed for adjoining vertices ($x_1,x_2,...,x_n$) **otherwise**.

Here again, the **Interpolation** function is an arbitrary one : it can be a n-linear one using the 2^n nearest nodes, a n-cubic one, or again, any other function. This is just another parameter...

2.3-The n-dimensional fractal field : for each point ($x_1,x_2,...,x_n$) of $\mathcal{E}$ we define a n-dimensional fractal field $\mathcal{F}$:

$$\mathcal{F}(x_1,x_2,...,x_n,S) = \sum_k p(k).f_k(x_1,x_2,...,x_n,S)$$

where p**(k)** is a ponderation factor such that :

$$p(1) > p(2) > ... > p(k) > ... > p(m).$$

The way p**(k)** is evaluated is **arbitrary**, but can be chosen, for example, proportional to the volume of the elementary cell of the mesh $\mathcal{M}_k$.

3-ANIMATION OF FRACTAL OBJECTS :

Let $\mathcal{S}$ be the physical space ; it is a subset of $\mathbf{R}^4$ (or $\mathbf{R}^3$, according to the dimension of the simulation) with coordinates (x,y,[z,]t). With these definitions, let's give two examples of the animation of fractal objects :

3.1-Earthquakes : it suffice to use the preceding model with the following **mapping** between $\mathcal{S}$ (= $\mathbf{R}^3$) and $\mathcal{E}$:

$$\begin{pmatrix} x \rightarrow x_1 \\ y \rightarrow x_2 \\ t \rightarrow x_3 \end{pmatrix}$$

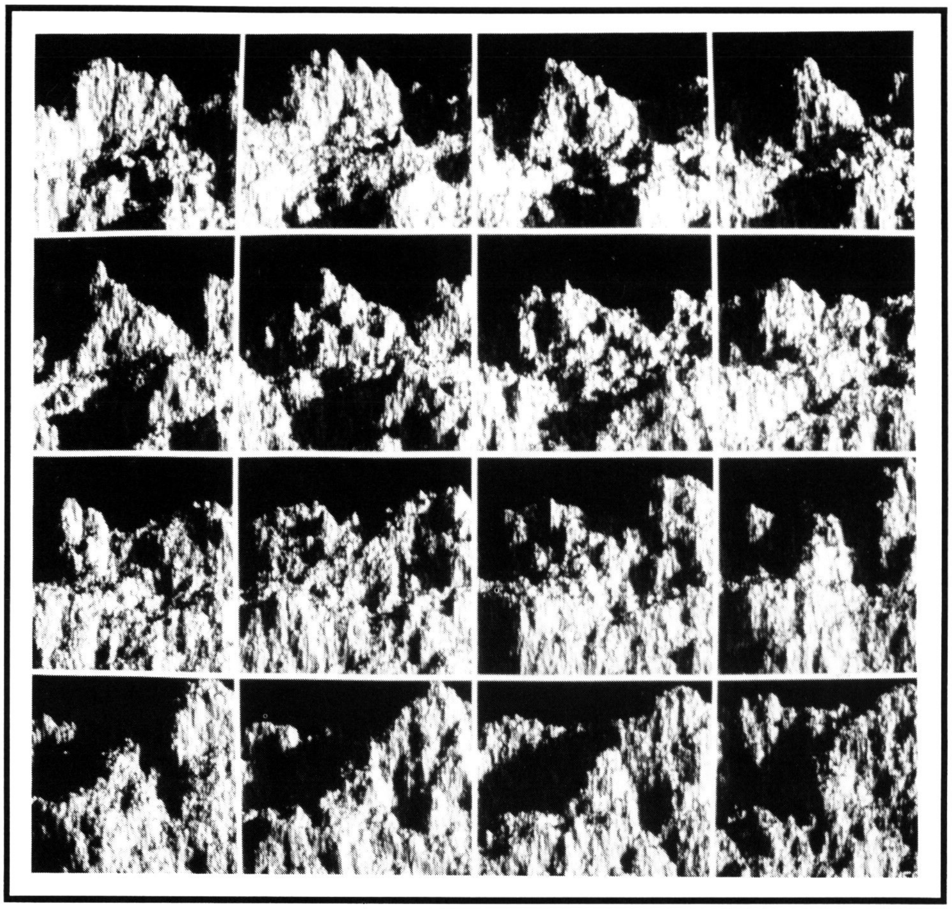


Figure 5 : 16 frames from a simulation of a catastrophic earthquake (each picture is 256x256, 256 colors).

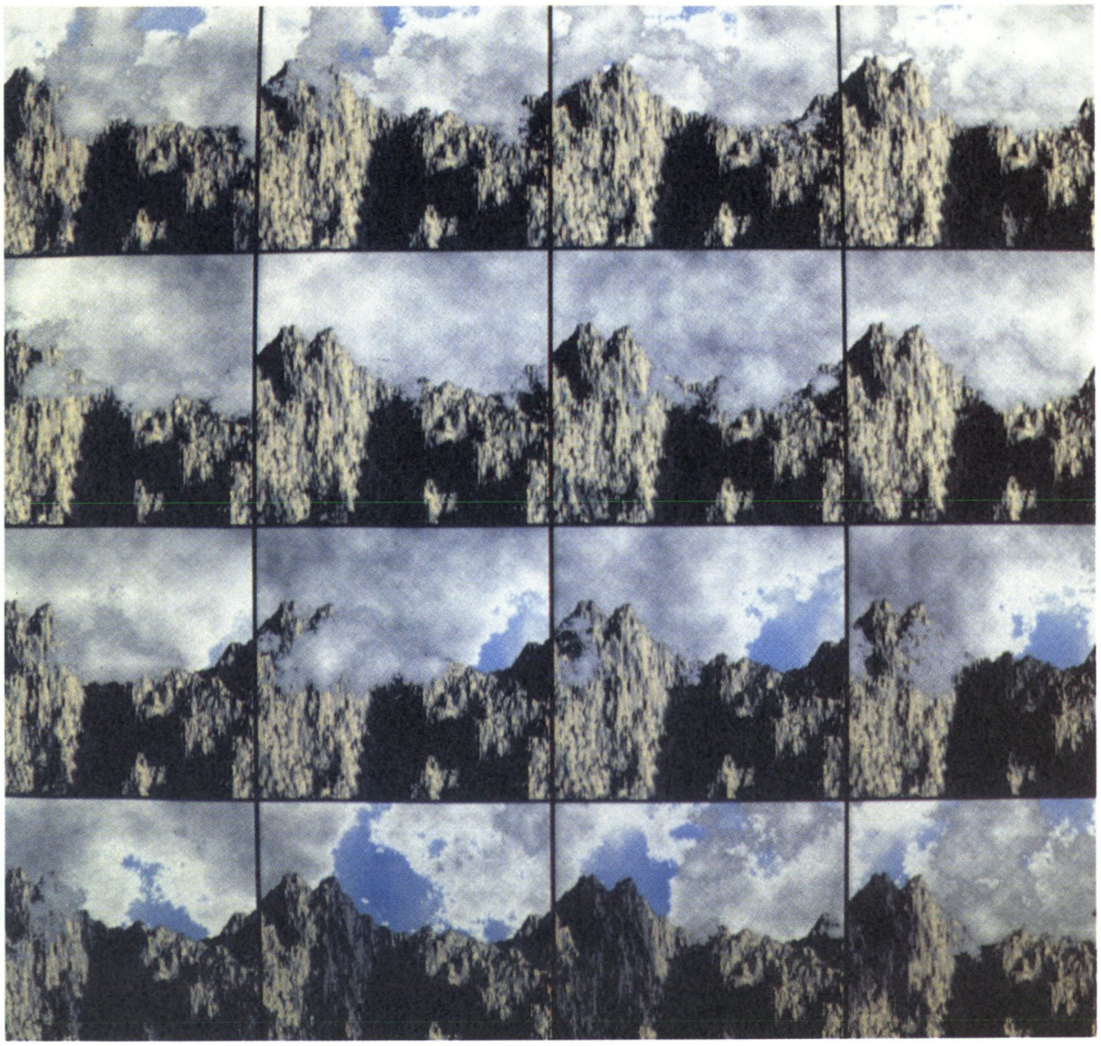


Figure 6 : 16 frames from a periodical (ie. the 17th picture would be identical to the first one) simulation of cloud dynamics ; the surface of the mountain is obtained by the mean of the two-dimensional fractal field shown on the figure 2 (each picture is 256x256, 256 colors).

The figure 5 shows sixteen frames from a terrific (because of the amplitude...) earthquake simulation. Each of the "instaneous" mountains is obtained with a two-dimensional **cross-section** inside the three-dimensional fractal field at **$t=x_3$=constant** ; then this "sub-field" is visualized as a three-dimensional surface.

3.2-Cloud dynamics : it suffice to use the preceding model with the following **mapping** between $\mathcal{S}$ (= $\mathbf{R}^3$ to reduce the computing time) and $\mathcal{E}$:

$$\begin{vmatrix} x \rightarrow \left[x_1 + \delta x_1(x_3)\right]_{\text{modulo } L_x} \\ y \rightarrow \left[x_2 + \delta x_2(x_3)\right]_{\text{modulo } L_y} \\ t \rightarrow \left[x_3\right]_{\text{modulo } L_t} \end{vmatrix}$$

where the vector $(\delta x_1(x_3),\delta x_2(x_3))$ is used to simulate the effect of the wind in the (x,y) plane, and the **modulos** **Lx/Ly/Lt** to allow the generation of periodic sequences. The figure 6 shows us a complex scene : a fractal mountain obtained by the preceding method of superposition, with three two-dimensional fields of clouds with a wind blowing from the right to the left ($\delta x_1(x_3) = -a^2.x_3$, $\delta x_2(x_3) = 0$). This simulation was only three-dimensionaly made in order to reduce the computing time. About the shadows, the ones relating to the mountains are correct, when the ones relating to the clouds are simulated (due to their two-dimensionality...) with a texture mapping approach : at time t, a texture using the clouds at time $t+\Delta t$ ($\Delta t \ll L_t$) is computed and then mapped onto the mountain surface.

4-CONCLUSION : this method is general ; it allows the generation of n-dimensional fractal objects, where one of the dimension can be the time, thus giving animation capabilities. Two axis of research will be followed : the first one is about the parallelization of this method on a Transputer network. The second one is related to the exploration of new mapping functions and meshes to produce new classes of fractal objects.

References :

[1] A. Fournier, D. Fussel, L. Carpenter, Computer rendering of stochastic models, Communication of the ACM, 25, 6, june 1982, pages 371-384.

[2] G. Miller, The definition and rendering of terrain maps, Computer Graphics, ACM SIGGRAPH, 20, 4, august 1986, pages 39-48.

[3] JF. Colonna, M. Farge, L'expérimentation numérique assistée par ordinateur, La Recherche, 187, april 1987, pages 444-457.

[4] JF. Colonna, Visualization, Computer Graphic World, december 1987, pages 40-44.

[5] JF. Colonna, Picture synthesis : an essential tool for numerical experimentation, Computer Physics Communications, 49, 1988, pages 215-228.

FIRST APPENDIX : This algorithm is in fact a small component of a much more larger software written to facilitate the manipulation and the visualization of large sets of scientific data [3][4][5]. A set of **cpp** macros (giving birth to C or Fortran sources) is the programming language and UNIX, the operating sytem. Computing fractal objects as fields allows us to manipulate them with general tools. They are C-like functions, like **mountain**(param), and each has its counterpart at the Shell level : hundreds of "pipeable" commands are available. Thus it is valid to write (where 'param' denotes lists of parameters) :

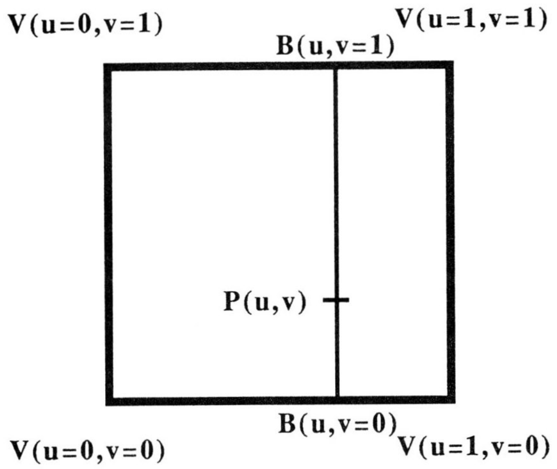
display(mountain(fractal_nD(param),param),param)**;**

or again, as a string of piped commands :

fractal_nD param | **mountain** param | **display** param

In both case, a n-dimensional fractal field is computed and displayed as a three-dimensional surface.

SECOND APPENDIX : a pseudo **n-cubic interpolation** will be described in a two-dimensional (n=2) space in order to simplify. Let $V_{(u=0,v=0)}$, $V_{(u=1,v=0)}$,$V_{(u=1,v=1)}$ and $V_{(u=0,v=1)}$ be the basic elements of the square mesh $\mathcal{M}_k$ and let **u** and **v** be the parametric coordinates of the square :



At the four vertices $\mathbf{V}_{(u,v)}$ (with **u** and **v** equal 0 or 1), we compute the value of the function $\mathbf{f}(\mathbf{V}_{(u,v)},\mathbf{k},\mathbf{S})$ as a random value :

$$\mathbf{f}(\mathbf{V}_{(u,v)},\mathbf{k},\mathbf{S}) = \mathbf{RDN}(u,v,k,S)$$

and the partial derivatives as finite differences :

$$\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{V}_{(u,v)},\mathbf{k},\mathbf{S}) = \frac{\mathbf{f}(\mathbf{V}_{(u+1,v)},\mathbf{k},\mathbf{S}) - \mathbf{f}(\mathbf{V}_{(u-1,v)},\mathbf{k},\mathbf{S})}{2}$$

$$\frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{V}_{(u,v)},\mathbf{k},\mathbf{S}) = \frac{\mathbf{f}(\mathbf{V}_{(u,v+1)},\mathbf{k},\mathbf{S}) - \mathbf{f}(\mathbf{V}_{(u,v-1)},\mathbf{k},\mathbf{S})}{2}$$

For the two points $\mathbf{B}_{(u,v)}$ (with v equal 0 or 1), we compute the value of the function **f** using a cubic interpolation with respect to u :

$$\mathbf{f}(\mathbf{B}_{(u,v)},\mathbf{k},\mathbf{S}) = \mathbf{a}_3.\mathbf{u}^3 + \mathbf{a}_2.\mathbf{u}^2 + \mathbf{a}_1.\mathbf{u}^1 + \mathbf{a}_0.\mathbf{u}^0$$

with :

$$\mathbf{a}_3 = (\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{V}_{(0,v)},\mathbf{k},\mathbf{S}) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{V}_{(1,v)},\mathbf{k},\mathbf{S})) - 2.(\mathbf{f}(\mathbf{V}_{(1,v)},\mathbf{k},\mathbf{S}) - \mathbf{f}(\mathbf{V}_{(0,v)},\mathbf{k},\mathbf{S}))$$
$$\mathbf{a}_2 = 3.(\mathbf{f}(\mathbf{V}_{(1,v)},\mathbf{k},\mathbf{S}) - \mathbf{f}(\mathbf{V}_{(0,v)},\mathbf{k},\mathbf{S})) - (2.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{V}_{(0,v)},\mathbf{k},\mathbf{S}) + \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{V}_{(1,v)},\mathbf{k},\mathbf{S}))$$
$$\mathbf{a}_1 = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{V}_{(0,v)},\mathbf{k},\mathbf{S})$$

$$\mathbf{a}_0 = \mathbf{f}(\mathbf{V}_{(0,v)},\mathbf{k},\mathbf{S})$$

and the partial derivative with respect to v, using a linear interpolation with respect to u :

$$\frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{B}_{(u,v)},\mathbf{k},\mathbf{S}) = \mathbf{a}_1.\mathbf{u}^1 + \mathbf{a}_0.\mathbf{u}^0$$

with :

$$\mathbf{a}_1 = \frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{V}_{(1,v)},\mathbf{k},\mathbf{S}) - \frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{V}_{(0,v)},\mathbf{k},\mathbf{S})$$
$$\mathbf{a}_0 = \frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{V}_{(0,v)},\mathbf{k},\mathbf{S})$$

At last, at the point $\mathbf{P}_{(u,v)}$, the value of the function is computed with a cubic interpolation with respect to v :

$$\mathbf{f}(\mathbf{P}_{(u,v)},\mathbf{k},\mathbf{S}) = \mathbf{a}_3.\mathbf{v}^3 + \mathbf{a}_2.\mathbf{v}^2 + \mathbf{a}_1.\mathbf{v}^1 + \mathbf{a}_0.\mathbf{v}^0$$

with :

$$\mathbf{a}_3 = (\frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{B}_{(u,0)},\mathbf{k},\mathbf{S}) + \frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{B}_{(u,1)},\mathbf{k},\mathbf{S})) - 2.(\mathbf{f}(\mathbf{B}_{(u,1)},\mathbf{k},\mathbf{S}) - \mathbf{f}(\mathbf{B}_{(u,0)},\mathbf{k},\mathbf{S}))$$
$$\mathbf{a}_2 = 3.(\mathbf{f}(\mathbf{B}_{(u,1)},\mathbf{k},\mathbf{S}) - \mathbf{f}(\mathbf{B}_{(u,0)},\mathbf{k},\mathbf{S})) - (2.\frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{B}_{(u,0)},\mathbf{k},\mathbf{S}) + \frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{B}_{(u,1)},\mathbf{k},\mathbf{S}))$$
$$\mathbf{a}_1 = \frac{\partial \mathbf{f}}{\partial \mathbf{v}}(\mathbf{B}_{(u,0)},\mathbf{k},\mathbf{S})$$

$$\mathbf{a}_0 = \mathbf{f}(\mathbf{B}_{(u,0)},\mathbf{k},\mathbf{S})$$

The recursive extension in a n-dimensional space of this pseudo n-cubic interpolation is obvious...