Stochastic Environmental Research and Risk Assessment (2020) 34:251–273
https://doi.org/10.1007/s00477-020-01770-8

ORIGINAL PAPER

Multiple-point statistics using multi-resolution images

Julien Straubhaar[1] · Philippe Renard[1] · Tatiana Chugunova[2]

Published online: 4 February 2020

Abstract
Multiple-point statistics (MPS) is a simulation technique allowing to generate images that reproduce the spatial features present in a training image (TI). MPS algorithms consist in sequentially filling a simulation grid such that patterns around the simulated values come from the TI. Following this principle, joint simulations of multiple variables can be handled and complex heterogeneous fields can be generated. However, inconsistent patterns are often found in the results and some spatial features can be difficult to reproduce. In this paper, a new MPS algorithm based on a multi-resolution representation of the TI is proposed to enhance the quality of the realizations. The method consists in first building a pyramid of images from the TI by successive convolution using Gaussian-like kernels. Secondly, a MPS simulation is done at the lowest resolution level. Then, the result is expanded to the next level of resolution (one rank higher) and used as a conditioning variable for a joint MPS simulation at that level. This last step is repeated up to the initial resolution, where the final simulation is retrieved. The method is implemented in the DeeSse code based on the direct sampling algorithm. Most of the features provided by the direct sampling (conditioning to hard data, uni- or multi-variate simulation of categorical and continuous variables, scaling and rotation of the training structures) are compatible with the proposed method and the usability is maintained. Finally, various examples show that in most of the situations, combining Gaussian pyramids with MPS allows to get results of better quality and in less time compared to direct MPS simulations.

Keywords Direct sampling · Gaussian pyramids · Convolution · Size reduction · Multi-variate simulations

1 Introduction

Multiple-point statistics (MPS) is a non-parametric method to generate random fields reproducing the spatial structures present in a conceptual model, the training image (TI). The principle is to borrow patterns from the TI to fill the simulation grid. MPS methods can be classified in two main groups: (1) the patch-based algorithms consisting in pasting several pattern nodes in the simulation grid at each step, such as *filtersim* (Zhang et al. 2006), *simpat* (Arpat and Caers 2007), *CCSIM* (Tahmasebi et al. 2012), *CIQ* (Mahmud et al. 2014), and techniques proposed by Rezaee et al. (2013) and Gardet et al. (2016; 2) the pixel-based algorithms, where only the central node is pasted, such as *snesim* (Strebelle 2002), *Impala* (Straubhaar et al. 2011, 2013), and the direct sampling (Mariethoz et al. 2010). Every method has their pros and cons, but whatever the method used, one encounters the common issue of simulations containing patterns not present in the TI. To reduce the number of incompatible patterns in the realizations, techniques have been developed such as post-processing (Strebelle and Remy 2005), real-time post-processing (Suzuki and Strebelle 2007), syn-processing (Mariethoz et al. 2010), or more recently a backtracking method (Shahraeeni 2019). These methods correct simulated values in the grid at the end or during the simulation process. Although these techniques are useful, it is important that the core of the simulation method produces realizations of sufficiently good quality. Thus, another idea to prevent as much as possible anomalies consists in adapting the TI before starting the simulation (Straubhaar et al. 2019).

✉ Julien Straubhaar
julien.straubhaar@unine.ch

[1] The Centre for Hydrogeology and Geothermics (CHYN), University of Neuchâtel, rue Emile-Argand 11, 2000 Neuchâtel, Switzerland

[2] Geostatistics and Uncertainties, TOTAL SA, 64000 Pau, France

Oriani et al. (2014) propose a method based on the direct sampling algorithm to simulate daily rainfall time series. They use in particular two auxiliary variables consisting in 2- and 365-day moving average (of the amount of rainfall) to guide the simulations and to better catch the complex structure of the times series. Indeed, with these variables, the statistics over different supports are used during the simulation process and this helps reproduce the signal.

In the same vein, our main motivation for the following work is to enhance MPS simulations by accounting for the characteristics of the TI at different scales. The rationale is that the spatial statistics depend on the support (the scale). Hence, we propose to integrate representations of the TI at different resolutions into a MPS scheme. The idea is to exploit the technique of data compression in image processing which consists in building Gaussian and Laplacian pyramids of the initial image (Burt and Adelson 1983). The Gaussian pyramid of an image is built by applying successive convolutions with Gaussian kernels. At each level, i.e. after each convolution, we obtain a smaller image in number of pixels providing a representation at a lower resolution. The Laplacian pyramid is built by expanding each image of the Gaussian pyramid. The expansion also consists in a moving average operation (convolution) and allows to retrieve an approximation of the image at a given level from the image available at the level one rank coarser.

Among the family of MPS algorithm, we focus on the direct sampling algorithm, introduced by Mariethoz et al. (2010), because it is one of the most flexible, allowing for joint simulations of multiple variables, it handles categorical as well as continuous variable, and does not require any data base of patterns (the TI is directly scanned for searching for patterns). We propose to combine the technique of pyramid with the direct sampling algorithm. The idea is to do a MPS simulation with the lowest resolution of the TI, and then successive joint simulations at every higher resolution level, using as TI the image from the Gaussian pyramid and the corresponding approximate image (obtained by expansion) and using the expanded image of the previous MPS simulation as conditioning variable. In this way, one catches the large scale spatial statistics first, which serves to guide the simulation up to the initial resolution.

This paper is organized as follows. In Sect. 2, both ingredients of the proposed method are presented: the direct sampling MPS algorithm (DeeSse) and the pyramid approach for building multi-resolution images. In Sect. 3, the new algorithm combining these two techniques is explained in detail, including various simulation set-ups such as conditioning data, categorical and continuous variables as well as non-stationarity. Section 4 is devoted to examples for demonstrating the applicability of the proposed method. Comparisons with classical direct sampling simulations are made and computational performance is discussed. Finally, a conclusion is given in Sect. 5.

2 Background on multiple-point statistics and Gaussian pyramids

2.1 Multiple-point statistics based on direct sampling

The direct sampling algorithm is presented in detail in Mariethoz et al. (2010). The simulation process consists in successively populating the cells in the simulation grid (SG) with values borrowed from the training image (TI). The basic idea is to select, at each iteration, a location in the TI where the surrounding data is similar to what is already simulated in the SG. Such a location is searched by directly sampling the TI. This very simple scheme allows for multi-variate simulations, i.e. joint simulations of several categorical or continuous variables, reproducing the spatial statistics within and between the variables given in the conceptual model.

The simulation algorithm relies on the comparison of patterns (in the SG and the TI) by using a distance function consisting of the proportion of mismatching nodes for a categorical variable and the mean absolute error for a continuous variable. The three key parameters required by the user are: the maximal number of nodes (also called neighbors) N in a pattern, an acceptation threshold t, which corresponds to the maximal distance for compatible patterns, and a maximal fraction f of the TI grid cells that can be scanned for the simulation of one SG cell.

The DeeSse algorithm (Straubhaar 2019) is used in this paper. It is illustrated in Fig. 1 and its main steps are described below in the multi-variate case which is the one of interest for this paper. Note that this algorithm is slightly different from the one described in Mariethoz et al. (2010) which aggregates the distances in a multi-variate context.

Let x and y denote a cell in the SG and in the TI grid respectively, and let $Z^{(k)}$ be the k-th variable considered on both grids. While it exists a cell in the SG with an uninformed variable, repeat:

(1) Choose a variable k and a cell x in the SG such that $Z^{(k)}(x)$ is unknown.
(2) For each variable $Z^{(j)}$: retrieve the pattern $d_j(x)$ in the SG, centered on x and constituted of the at maximum N_j closest informed nodes for that variable.
(3) Set the global error E_{best} to infinity and scan the TI grid cells y randomly:

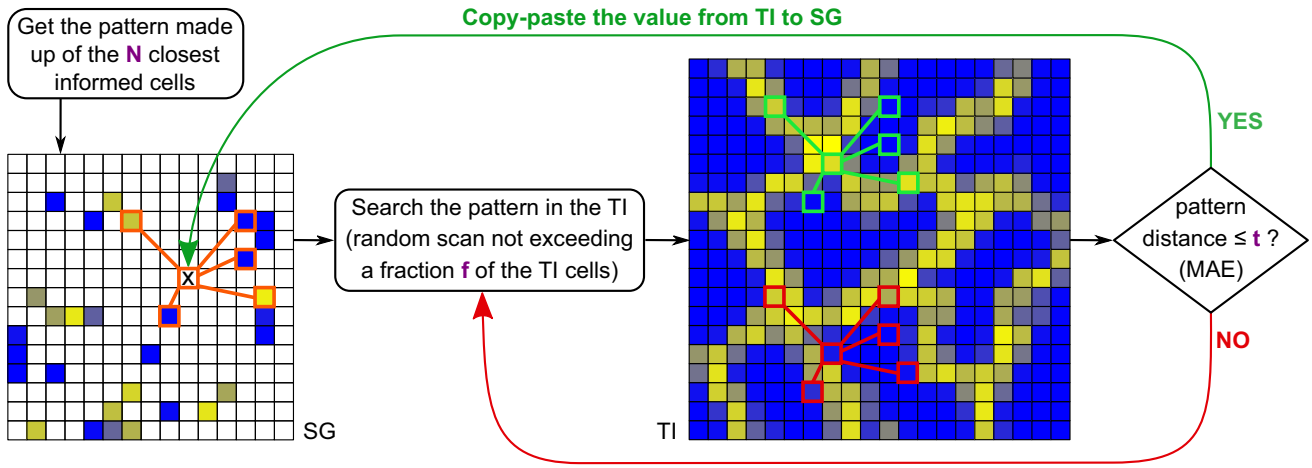

Fig. 1 MPS by direct sampling (DeeSse algorithm): illustration of the algorithm for the simulation of one cell (uni-variate case)

(3a) Get the pattern $d_j(y)$ centered on y in the TI for every variable $Z^{(j)}$, the pattern geometries being defined by the patterns retrieved from the SG.

(3b) For every variable, compute a relative error

$$E_j(y) = \max\left\{\frac{D_j\big(d_j(x), d_j(y)\big) - t_j}{t_j}, 0\right\} \quad (1)$$

where $D_j(\cdot,\cdot)$ is a distance function between patterns and t_j is the user defined acceptation threshold for the variable $Z^{(j)}$.

(3c) Compute the global error $E(y) = \sum_j E_j(y)$ and if $E(y) < E_{best}$, set $y_{best} = y$.

(3d) If $E_{best} = 0$, i.e. the distance between the patterns in the SG and in the TI is less than or equal to the acceptance threshold for every variable, or if a fraction f of the TI has been sampled, exit the $(3a{-}d)$-loop.

(4) Assign the value $Z^{(k)}(y_{best})$ to $Z^{(k)}(x)$ for the selected k, or alternatively for all variables k such that $Z^{(k)}(x)$ is uninformed.

The distance D_j between two patterns in the Eq. (1) is set according to the type of the j-th variable. For a categorical variable, the distance is simply the proportion of mismatching cells between the two patterns, i.e the Hamming distance normalized by the pattern size. For a continuous variable, the mean absolute error (of the values in the cells of the two patterns) is usually used. In this case, the variable Z is normalized before starting the simulation in the interval [0, 1] by the application $Z \longmapsto (Z - \min_{y\in TI} Z(y))/(\max_{y\in TI} Z(y) - \min_{y\in TI} Z(y))$, and then back transformed at the end of the simulation. Hence the distance between two patterns is comprised between 0 and 1 whatever the type of the variable, and the acceptance threshold can be interpreted as an error proportion.

Note also and finally that this algorithm is parallelized for shared memory machines. When running on multiple threads (CPUs), one grid cell is simultaneously simulated by each thread, provided that the patterns centered on the simulated cells would be exactly the same ones if only one thread was used (based on the same random simulation path), which ensures the reproducibility of the results.

2.2 Multiple-resolution images with Gaussian pyramids

The Gaussian pyramids technique allows to build a sequence of images G_j $(j = 1, 2, \ldots)$ of reduced sizes (lower resolutions) of an initial image G_0, by successively applying a convolution (moving average) with a Gaussian-like kernel (Burt and Adelson 1983). In its standard implementation, the number of cells in each direction is divided by a factor 2 for each pyramid level. Here, we propose a generalization to a reduction factor k (integer). Note that averaging is meaningful only for continuous variables or binary indicator variables.

2.2.1 Reduce and expand operations

As these operations can be applied successively for each direction, one can consider only the uni-dimensional case. The *reduce operation*, RED_k, allows to obtain the image $G_{j+1} = RED_k(G_j)$ from the image at the previous level. The value in the cell of index i in the image G_{j+1} is defined as the weighted mean

$$G_{j+1}(i) = \sum_{l=-k}^{k} \omega_l \cdot G_j(k \cdot i + l). \tag{2}$$

This is a convolution with a kernel of width $2k+1$. The weights $(\omega_{-k}, \ldots, \omega_k)$ are defined such that: (i) they are symmetric ($\omega_{-l} = \omega_l$ for any l), (ii) they are positive and sum to one, and (iii) the contribution of each cell to the reduced image is the same. This implies

$$\omega_0 + 2\omega_k = \omega_l + \omega_{k-l} = \frac{1}{k}, \tag{3}$$

for $l = 1, \ldots, k-1$. To get a Gaussian shape kernel, one uses the binomial coefficients $\binom{2k}{l}$ and set

$$\{\omega_0, \omega_k\} = \frac{1}{k} \cdot \left[\binom{2k}{k} + 2\right]^{-1} \cdot \left\{\binom{2k}{k}, 1\right\} \tag{4}$$

and

$$\{\omega_l, \omega_{k-l}\} = \frac{1}{k} \cdot \left[\binom{2k}{k-l} + \binom{2k}{l}\right]^{-1} \cdot \left\{\binom{2k}{k-l}, \binom{2k}{l}\right\}, l = 1, \ldots, k-1 \tag{5}$$

according to the conditions (i-iii) above. Note that for the classical Gaussian pyramid, $k = 2$ and the weights are proportional to the binomial coefficients, $\{\omega_{-2}, \omega_{-1}, \omega_0, \omega_1, \omega_2\} = 1/16\{1, 4, 6, 4, 1\}$.

As the number of cells decreases after applying the reduce operation, some information is lost and the original image G_j cannot be recovered from the reduced image G_{j+1}. However, an approximation can be computed by applying the pseudo-inverse *expand operation*, $\widetilde{G}_j = EXP_k(G_{j+1})$, defined as

$$\widetilde{G}_j(i) = k \cdot \sum_{l=-k}^{k} \omega_l \cdot G_{j+1}\left(\frac{i-l}{k}\right), \tag{6}$$

where only the terms such that $(i-l)/k$ is integer are considered in the sum. Both reduce and expand operations are illustrated in Fig. 2 for the case $k = 3$.

2.2.2 Managing the image borders

The Eqs. (2) and (6) defining the *RED* and *EXP* operations do not take care of the borders of the images. Let start with an image G_j containing M_j cells (indexed from 0 to $M_j - 1$). Both operations are illustrated in Fig. 3 for a reduction factor of $k = 3$ and $M_j = 9$. Consider the Euclidean division of $M_j - 1$ by k, $M_j - 1 = q \cdot k + r$, where q is the quotient and r the remainder. Then, the reduced image $G_{j+1} = RED_k(G_j)$ is of size

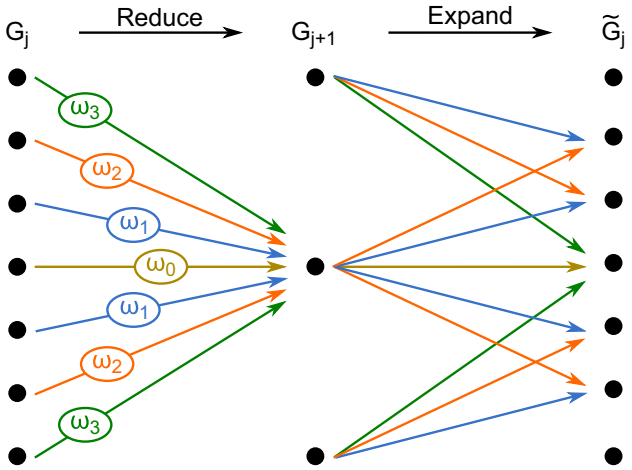

Fig. 2 Gaussian pyramids—illustration of reduce and expand operations in one dimension for a factor $k = 3$; the grid cells are represented by black dots aligned vertically; both operations consist in a moving average with the weights ω_i for the reduce operation (left), and the weights set according to the colors of the arrows and multiplied by k for the expand operation (right)

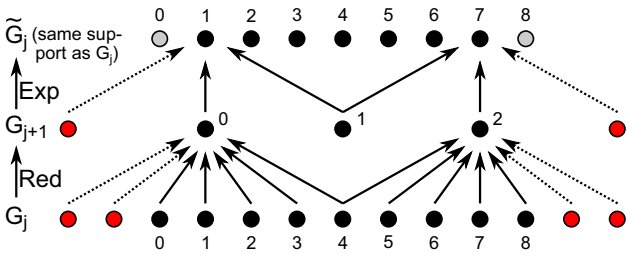

Fig. 3 Gaussian pyramids—illustration of reduce and expand operations in a uni-dimensional grid of 9 cells for a factor $k = 3$; the grid cells are indexed and represented by dots aligned horizontally; red cells are temporarily "added" (with value copied from the nearest cell), gray cells are added with a missing value such that the images G_j and $\widetilde{G}_j$ have the same dimensions

$$M_{j+1} = 1 + \left\lfloor \frac{M_j - 1}{k} \right\rfloor = 1 + q \tag{7}$$

where $\lfloor x \rfloor$ denotes the greatest integer less than or equal to x, and the size of the expanded image $\widetilde{G}_j = EXP_k(G_{j+1})$ is expressed as

$$\widetilde{M}_j = 1 + k \cdot (M_{j+1} - 1) = M_j - r. \tag{8}$$

To compute the value attached to a cell in a border of the reduced (resp. expanded) image, at most k cells (resp. exactly 1 cell) are/is temporarily added in the input image such that the filter (kernel) is entirely filled. The value in the added cells (red cells in Fig. 3) is simply copied from the nearest grid cell. Note that r cells are "lost" from G_j to $\widetilde{G}_j$. However, one can consider the image $\widetilde{G}_j$ of the same size (same support) as image G_j by adding r cells with a *missing value* (unknown value), splitted in two balanced

sets in the borders of the grid (gray cells in Fig. 3). This can be done by adding s cells at the left border (i.e. border with small cell indexes) and $r - s$ at the opposite border, with s the quotient of the Euclidean division of r by 2. For example, $r = 2$ and $s = 1$ in the case of Fig. 3. Note that accounting for the borders in this way requires to adapt the reduce and expand operations by integrating the possible shift s: the index $k \cdot i + l$ is replaced by $s + k \cdot i + l$ and $(i - l)/k$ by $(i - s - l)/k$ in the right-hand side of the Eqs. (2) and (6) respectively.

Gaussian pyramids are illustrated for a bi-dimensional binary image, using a reduction factor $k = 2$ and $k = 3$ in Figs. 4 and 5 respectively. The original image represents channels at two different scales, the main (larger) channels are emphasized at lower resolution in the pyramid levels. Note that the reduction factors could be chosen independently for each direction and each level transition. These factors are indicated with index notation, hence, $RED_{(k_x,k_y)}$ (resp. $EXP_{(k_x,k_y)}$) denotes the reduce operation (resp. expand operation) applied to a bi-dimensional image with the factors $k = k_x$ in the x-axis direction and $k = k_y$ in the y-axis direction.

3 MPS based on pyramids: the methodology

In this section, we show how Gaussian pyramids (Sect. 2.2) can be integrated in the DeeSse algorithm (Sect. 2.1) in order to better account for structures at different scales. The general idea is to consider a multi-resolution TI built using the Gaussian pyramids technique, and perform hierarchical conditional MPS simulations, from the lowest resolution to the highest one.

3.1 Simulation of a continuous or binary variable

The methodology is described here in the case where the variable is continuous or binary. In this situation, the Gaussian pyramids technique can be directly applied onto the considered variable as computing moving averages makes sense. The flowchart of the algorithm in this case is displayed in Fig. 6 and explained in detail in the following.

3.1.1 Building the pyramid

First, one specifies the number L of pyramid levels additional to the initial resolution, and the reduction factors defining the reduce (resp. expand) operations $RED^{(j)}$ (resp. $EXP^{(j)}$) from the level j to the level $j+1$, for $j = 0, \ldots, L-1$, the levels being indexed from 0, the initial and highest / finest resolution, to L, the last and lowest / coarsest resolution.

The initial step consists in normalizing the TI variable and then building the pyramid:

$$TI_0 = TI, \tag{9}$$

$$TI_{j+1} = RED^{(j)}\left(TI_j\right), \tag{10}$$

$$\widetilde{TI}_j = EXP^{(j)}\left(TI_{j+1}\right), \tag{11}$$

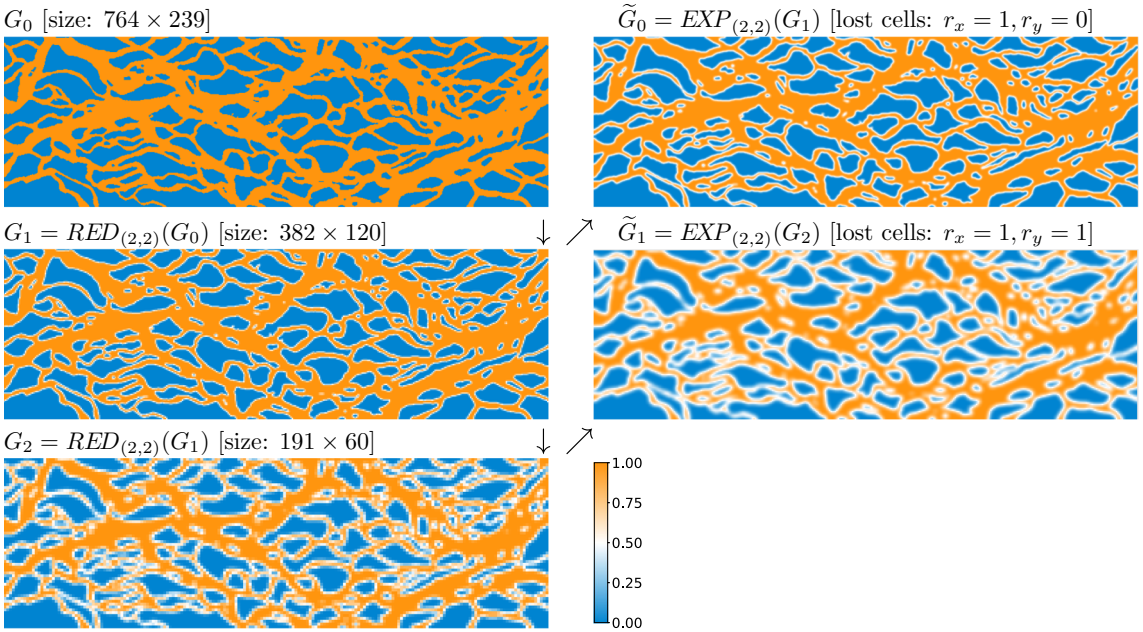

Fig. 4 Gaussian pyramids applied to an initial bi-dimensional binary image (top left), with $L = 2$ levels additional to the initial resolution and a reduction factor of $k = 2$; the dimensions of the grids and the number of "lost" cells (gray cells) in each direction are indicated for each level

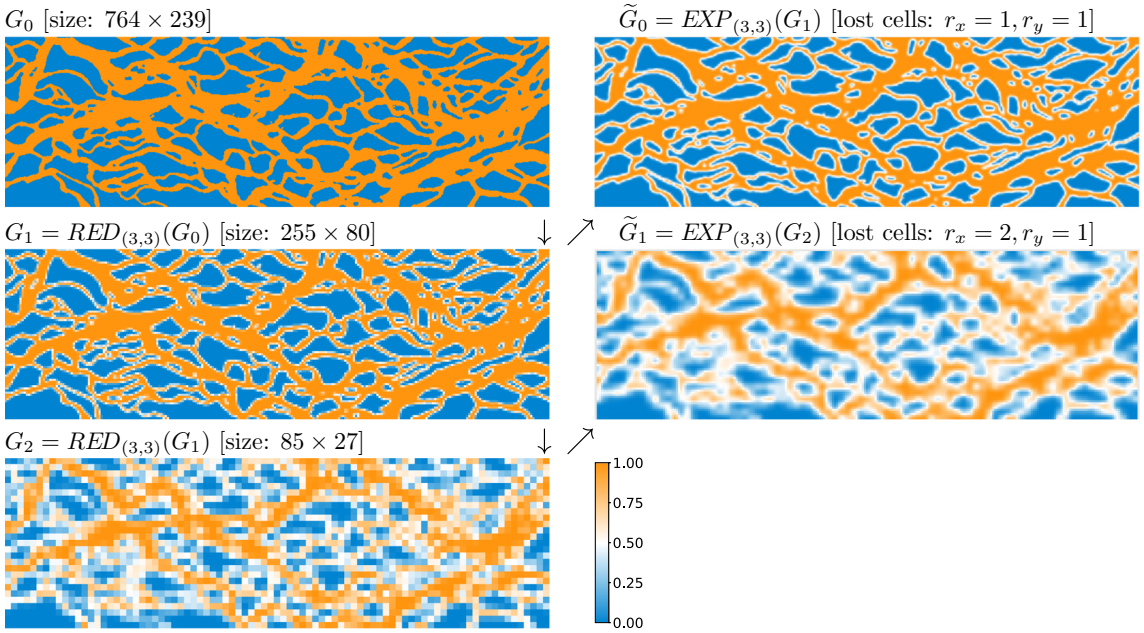

Fig. 5 Gaussian pyramids applied to an initial bi-dimensional binary image (top left), with $L = 2$ levels additional to the initial resolution and a reduction factor of $k = 3$; the dimensions of the grids and the number of "lost" cells (gray cells) in each direction are indicated for each level

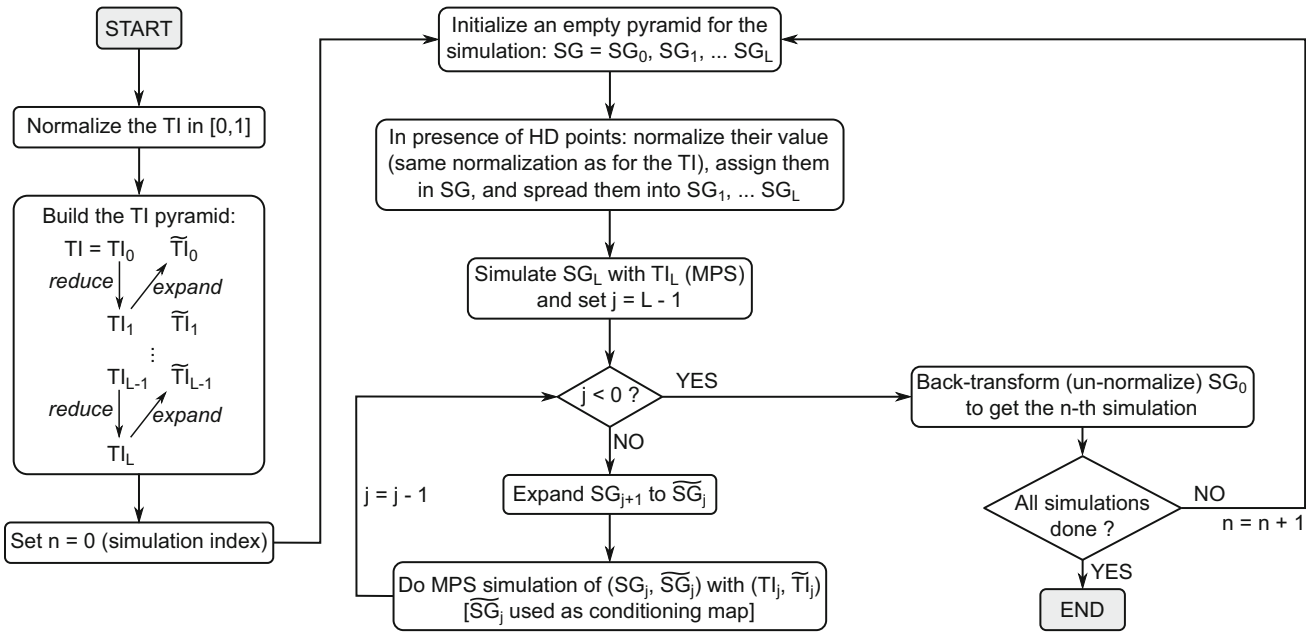

Fig. 6 Flowchart of the multi-scale MPS algorithm for a continuous or binary variable

for $j = 0, \ldots L-1$.

For the simulation grid, one considers a pyramid $SG_0, SG_1, \ldots, SG_L$, and $\widetilde{SG}_0, \ldots, \widetilde{SG}_{L-1}$, initialized with uninformed cells, i.e. a missing value in any cell, of the same resolution at each level as in the pyramid for the training image. Note that in a level j, the images TI_j and $\widetilde{TI}_j$ (resp. SG_j and $\widetilde{SG}_j$) are defined on the same grid, assuming that applying a reduce operation and then an expand operation keeps the support unchanged by considering uninformed cells in the borders of the expanded image (corresponding to "lost" cells) if needed (see end of Sect. 2.2.2 and Fig. 3).

3.1.2 Simulation using the pyramid

The main steps to generate a realization are the following ones.

(1) Assign conditioning data (if any) into SG_0, and spread conditioning information to $SG_1, \ldots, SG_L$ (see Sect. 3.2).
(2) Do a MPS simulation at level L (in SG_L) using DeeSse and the training image TI_L.
(3) For $j = L-1, \ldots, 0$, successively do:
  (3a) Expand the image simulated at level $j+1$: $\widetilde{SG}_j = EXP^{(j)}(SG_{j+1})$.
  (3b) Do a MPS simulation of SG_j at level j conditionally to $\widetilde{SG}_j$. DeeSse is used in bivariate mode with the vector of variables $(SG_j, \widetilde{SG}_j)$ for the simulation grid and $(TI_j, \widetilde{TI}_j)$ as the training image.

In step (3b), the secondary variable is obtained from the expansion of the primary variable at the pyramid level one rank coarser. It is exhaustively known except for a few possible cells in the borders (called the "lost" cells in Sect. 2.2.2).

Each DeeSse simulation requires as input parameters the maximal numbers of neighbors (N) in the search patterns and the acceptation threshold (t) for each variable, plus the maximal scanned fraction (f) (Sect. 2.1). We propose to derive these parameters for the simulation at each level from a unique set N, t, f to avoid requesting to too many parameters from the user. As a consequence, using pyramid will require as additional parameters only L (the number of levels additional to the initial resolution), and k (the reduction factor(s) for the level transitions).

3.1.3 MPS parameters in the pyramid

The maximal scanned fraction at the level j is set to

$$f_j = \min\left(f_{max}, \frac{S}{S_j} \cdot f\right), \tag{12}$$

where S is the size (number of cells) of the initial TI and S_j the size of the TI at the level j, and f_{max} is a maximal value, constant typically set to 1 or 0.9 to avoid an exhaustive scan. Thus, the maximal number of scanned cells at the level j does not exceed $f_j \cdot S_j \leqslant f \cdot S$.

As the TI values are normalized only for the initial level (0), the acceptation threshold is adapted at the level j according to the new range of values. It is defined, for any level j, as

$$t_j = \left(\max_{y_j \in TI_j} Z(y_j) - \min_{y_j \in TI_j} Z(y_j)\right) \cdot t, \tag{13}$$

$$\tilde{t}_j = \left(\max_{y_j \in \widetilde{TI}_j} Z(y_j) - \min_{y_j \in \widetilde{TI}_j} Z(y_j)\right) \cdot t, \tag{14}$$

for the first variable (simulated) and the second variable (expanded from the level one rank coarser) respectively.

The idea for the maximal number of neighbors is to decrease it as one moves to a finer level. The two main reasons are: (1) the simulation being conditioned by the expanded variable, it is well-guided and then reduced pattern are sufficient to reproduce the spatial features, and (2) as the resolution becomes finer, the total number of cells to simulate increases and reducing the size of the pattern allows to speed up the simulation. Moreover, the second (expanded) variable being exhaustively known, a smaller number of nodes in the pattern than for the first (simulated) variable is sufficient to properly filter out compatible regions in the training image. We propose the following set-up. For the primary variable, the maximal number of neighbors is divided by 2 only twice (so that this number does not become too small, which could affect the reproduction of the spatial structures),

$$N_L = N,\ N_{L-1} = \left\lfloor \frac{N}{2} \right\rfloor,\ N_j = \left\lfloor \frac{N}{4} \right\rfloor,\ \text{for } L-2 \geqslant j \geqslant 0, \tag{15}$$

and, for the second (expanded) variable,

$$\widetilde{N}_j = \left\lfloor \frac{N_j}{3} \right\rfloor,\ \text{for any } j. \tag{16}$$

Note also that one can impose that the maximal number of neighbors does not fall below a prescribed minimal value for any case.

3.1.4 Illustrative example of unconditional simulations with pyramids

The proposed method is applied with the previous TI, $L = 2$ pyramid levels additional to the initial resolution and a same reduction factor k for x and y directions and for every level transition. The three main parameters for DeeSse are set to $N = 24$ (maximal number of neighbors), $t = 0.02$ (acceptation threshold) and $f = 0.33$ (maximal scanned fraction), and adapted for each pyramid level as described above. The results obtained with the reduction factor $k = 2$ and $k = 3$ are displayed respectively in Figs. 7 and 8. Note that the top left image in Fig. 4 (resp. Fig. 5) is the given TI and the top left image in Fig. 7 (resp. Fig. 8) is the final result obtained by the algorithm sketched in Fig. 6.

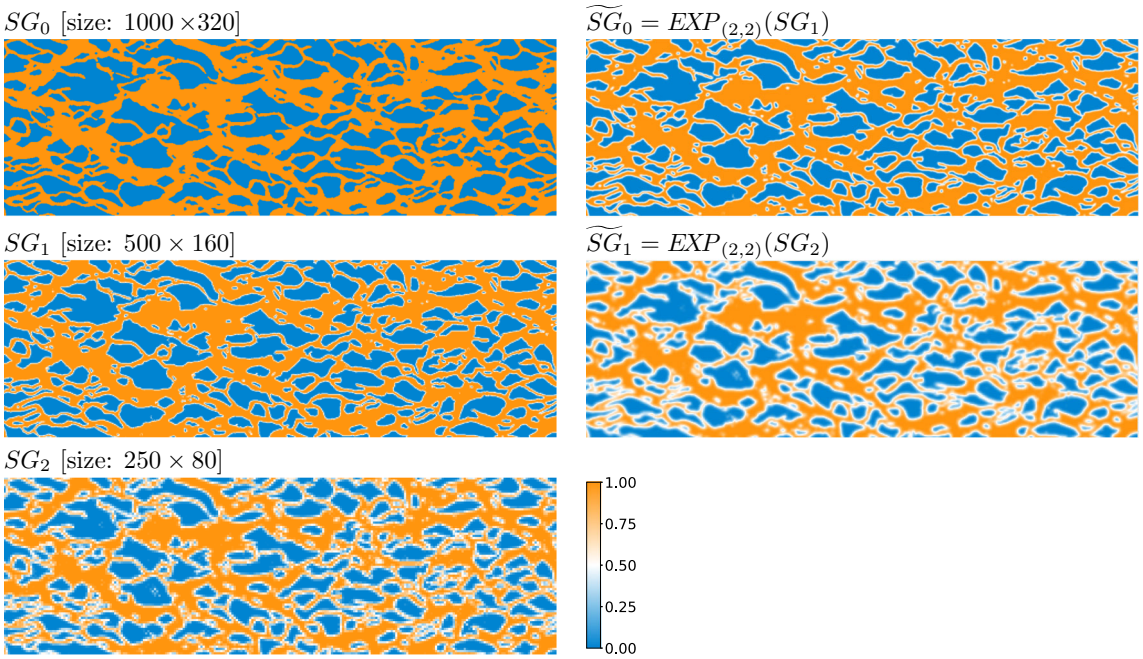

Fig. 7 MPS simulation (unconditional) using pyramids with $L = 2$ (number of levels), and $k = 2$ (reduction factor), based on the TI of Fig. 4. Results at each level are displayed

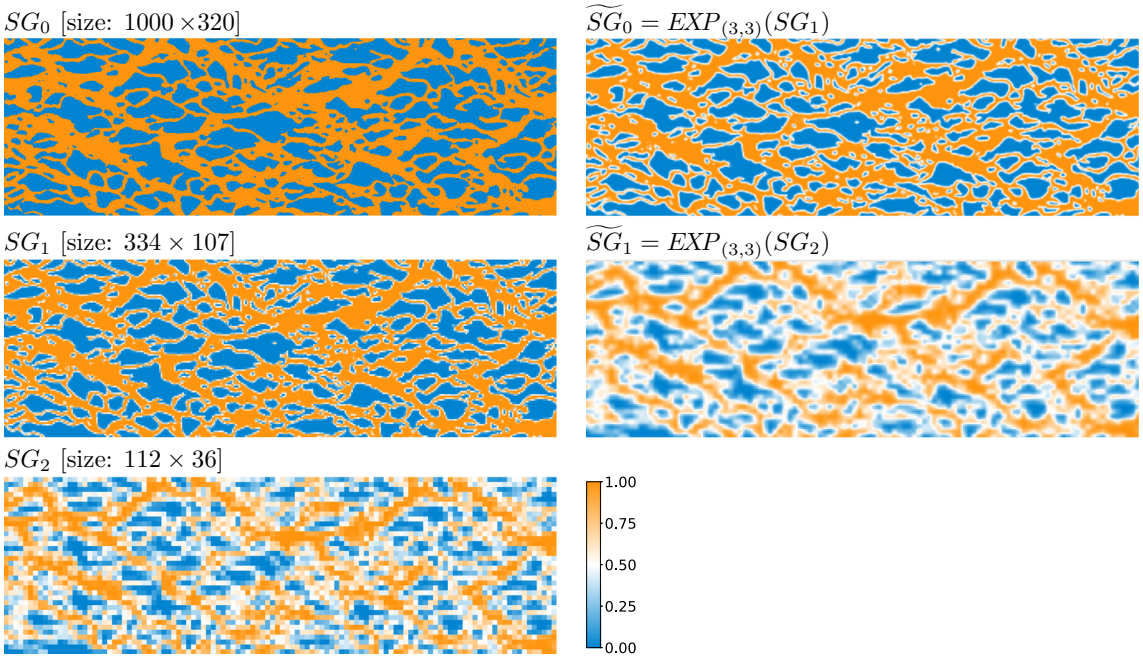

Fig. 8 MPS simulation (unconditional) using pyramids with $L = 2$ (number of levels), and $k = 3$ (reduction factor), based on the TI of Fig. 5. Results at each level are displayed

3.2 Accounting for conditioning data with pyramids

Consider a set of conditioning data, also called hard data, which consists in a set of points located in the simulation grid and with known values for the simulated variable. Such data is related to the original grid, and the value of each data point is assigned to the cell in SG_0 containing that point.

As the simulation follows a hierarchical sequence starting from the lowest resolution grid (SG_L) to the the finest resolution grid (SG_0), it is blind to the conditioning

cells lying in SG_0 when simulating coarser levels. As mentioned earlier, to tackle this hindrance we propose a preliminary process that consists in spreading the conditioning cells through each pyramid level. Note that the conditioning information needs to be brought only in the "reduced" images SG_j (as primary variable), since the expanded images ($\widetilde{SG}_j$, secondary variable) are computed during the simulation process.

Let us first consider two situations to explain the main ideas to propagate hard data information. On the one hand, assume that conditioning points cover entirely a contiguous area of the simulation grid. In this situation, the hard data values can simply be propagated through the pyramid by applying the reduce operation to this area. On the other hand, consider an isolated conditioning point in the simulation grid. In that case, one can not proceed as before, because the values of the variable around that point are not known. The idea is therefore to use MPS between two successive pyramid levels, and simulate the variable in some cells in the coarser grid conditioned to the finer grid containing the conditioning cells. As the two involved grids do not have the same size (change of support), the classical MPS technique needs to be adapted. Following this idea, information from sparse data is stochastically transfered from one level to the next, therefore accounting for uncertainty.

Based on these two situations, we develop a method accounting for any conditioning data set and consisting in the two successive steps: (1) deterministic propagation (Sect. 3.2.1), and (2) stochastic propagation (Sect. 3.2.2). An illustrative example is given in Sect. 3.2.3.

3.2.1 Step 1: deterministic propagation of (dense) hard data

First, every cell in $SG_0, \ldots, SG_L$ is initialized with a missing value. The hard data points are assigned in SG_0. Then, the reduce operation $RED^{(j)}$, for $j = 0, \ldots, L-1$ is successively applied, while accounting for the uninformed cells in the following manner. A reduced value is computed as a weighted average over cells from the grid at the previous level, by ignoring missing values (uninformed cells). If the sum of the weights over the informed locations are greater than or equal to a given constant C, the weights are re-normalized and the weighted average value is computed and set into the output cell, otherwise the output cell is left as uninformed. In this paper, the threshold value $C = 0.6$ is used.

Following this step, dense informed regions in the original level SG_0 are spread *in a deterministic way* through the pyramid. As this process is deterministic, this step is done only once whatever the number of realizations.

3.2.2 Step 2: stochastic propagation of (sparse) hard data

After the step 1 above, the second step is applied successively for $j = 0, \ldots, L-1$. A cell x_{j+1} in SG_{j+1} corresponds to the cell x_j in SG_j on which the filter of the reduce operation would be centered. Let W_{j+1} be the ensemble of cells x_{j+1} that are uninformed and for which the filter of $RED^{(j)}$ centered at the corresponding cells x_j covers at least one informed cell in SG_j. Then, the cells x_{j+1} in W_{j+1} are successively simulated, in a random order, and conditionally to the previous level. The simulation is done following the direct sampling strategy, i.e. the patterns centered on x_{j+1} and on the corresponding node x_j are retrieved, then the cells y_{j+1} in TI_{j+1} are randomly scanned and the patterns centered on y_{j+1} and on the corresponding cell y_j in TI_j are compared to those in the pyramid of the SG. The maximal scanned fraction f_{j+1} and the acceptation thresholds t_{j+1} and t_j are set according to the level they refer to as given in Eqs. (12) and (13), whereas the maximal numbers of neighbors N_j and N_{j+1} are defined as the half of N.

As this step involves simulations (stochastic process), it is done for every new realization. Hence, the cells in the ensembles W_j, $j = 1, \ldots L$, called also weak conditioning cells, are populated by different values through the ensemble of realizations.

3.2.3 Illustrative example of conditional simulation with pyramids

The 2-step process described above is illustrated by an example in Figs. 9 and 10. We consider the same TI and the same DeeSse parameters as for the unconditional case above (Sect. 3.1.4), and a conditioning data set (top left image in Figs. 9 and 10) formed by two areas densely informed at left and right in the grid, a 1-cell-width line in-between, and 6 isolated points. The left column in these figures shows the pyramid of the SG after the propagation of conditioning information, and the right column the result of the simulation at the end. Note that the "expanded" images in the pyramid are not shown, and that the final result corresponds to the top right image.

3.3 Simulation of a categorical variable with pyramids

In the two previous sections (Sects. 3.1, 3.2), MPS simulation with pyramids has been presented in the case of a continuous or binary variable. As the pyramids rely on moving averages, they have to be applied to this type of variable.

In the case of a multi-category (multi-facies) discrete variable, the category values are abstract codes and do not

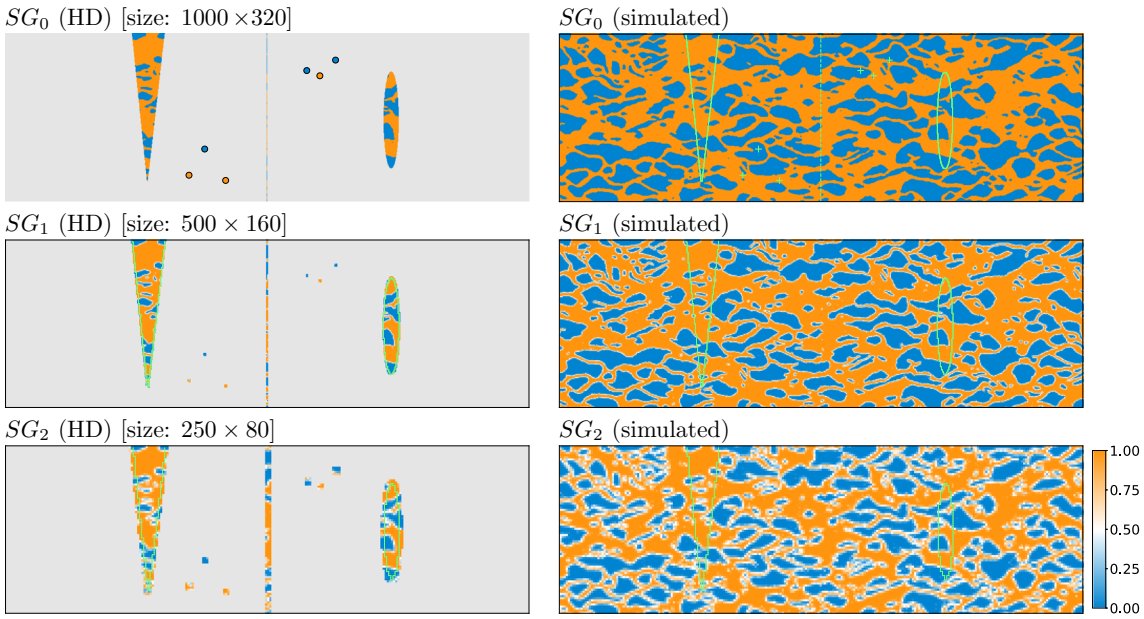

Fig. 9 MPS conditional simulation using pyramids with $L = 2$ (number of levels), and $k = 2$ (reduction factor), based on the TI of Fig. 4. Left column) pyramid of the SG after dealing with hard data (HD): in top image the 6 isolated HD points are exaggerated, in two other images the green lines delineate cells informed after step 1 (deterministic); right column) results at each level

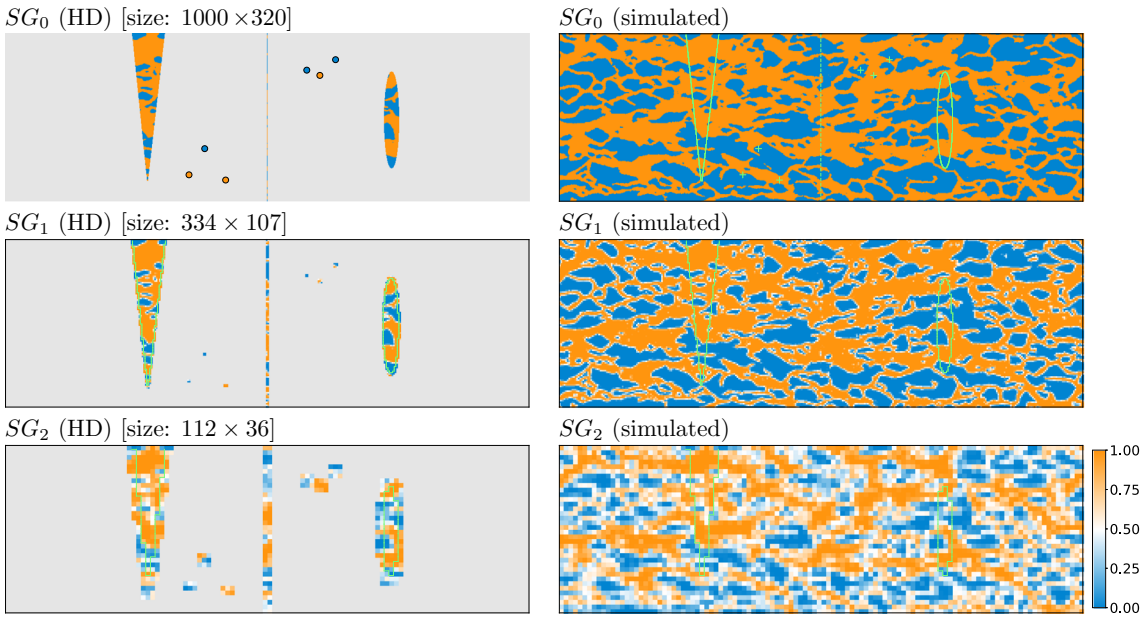

Fig. 10 MPS conditional simulation using pyramids with $L = 2$ (number of levels), and $k = 3$ (reduction factor), based on the TI of Fig. 5. Left column) pyramid of the SG after dealing with hard data (HD): in top image the 6 isolated HD points are exaggerated, in two other images the green lines delineate cells informed after step 1 (deterministic); right column) results at each level

correspond to ordered values, then computing moving averages directly does not make sense. In the two next sections, we propose two ways to extend the methodology for categorical variables.

3.3.1 Pyramids for the indicator variables of the categories

The first way to deal with a categorical variable is to build one pyramid for the indicator variable of each category (except one). Consider a n-category training image, with categories $c_1, \ldots, c_n$, and let TI be the original image, $TI^{(k)}$ the binary image provided by the indicator of category c_k,

and $TI_j^{(k)}$ (resp. $\widetilde{TI}_j^{(k)}$) the "reduced" (resp. "expanded") image representation of $TI^{(k)}$ at the pyramid level j. Then, for a simulation with L pyramid levels additional to the initial resolution, the considered images (variables) are

$TI, \widetilde{TI}_0^{(1)}, \ldots, \widetilde{TI}_0^{(n-1)}$ at level 0,
$TI_j^{(1)}, \ldots, TI_j^{(n-1)}, \widetilde{TI}_j^{(1)}, \ldots, \widetilde{TI}_j^{(n-1)}$ at level j, for $j = 1, \ldots, L-1$, and
$TI_L^{(1)}, \ldots, TI_L^{(n-1)}$ at level L.

For the simulation grid, the propagation of conditioning data information is done for the same indicator variables. Then the simulation procedure is similar to what was presented previously but more variables are involved at each level.

3.3.2 Pyramids for one representative continuous variable

Let $c_1, \ldots, c_n$ be the categories of the considered variable. The previous method based on the pyramids of the indicator variables of $n-1$ categories can become slow when n increases, because of the large number of variables to consider at each pyramid level. One idea to avoid that is to represent the original categorical image by one continuous image, and then use the pyramid of this latter image to guide the simulation. As a pyramid relies on moving averages, we propose to build a representative continuous variable such that close values correspond to most connected categories in the TI.

First, the TI is quickly analyzed to compute the $n \times n$ matrix $T = (t_{ij})$ counting the number of contacts / transitions from each pair of categories through adjacent cells. The coefficient t_{ij} is the number of times that two adjacent cells in the TI grid have the categories c_i and c_j. Two grid cells are considered adjacent if they share a face (or edge in two dimensions) orthogonal to an axis direction (x, y or z) involved in the moving average filter used to build the pyramid. The matrix T is symmetric. It is then used to re-order the list of categories as $\{c_{i_1}, \ldots, c_{i_n}\}$ as follows.

(1) The indices $k_1 \neq k_2$ corresponding to a maximal non-diagonal coefficient in T is retrieved, and the vector of indices $I = (k_1, k_2)$ is formed.
(2) Let $I = (k_1, \ldots, k_2)$ be the current vector of distinct indices of length m, $2 \leqslant m < n$. One searches for a coefficient in the matrix T realizing the maximum value among the non-diagonal coefficients having two distinct indices and whose one of them is equal to k_1 or k_2 and the other one not in the vector I. The index k not in I of that coefficient is then prepend to I if k_1 is the other index, or append to I if k_2 is the other index.
(3) The step (2) is repeated until the vector $I = (i_1, \ldots, i_n)$ contains the index 1 to n, and the corresponding list of re-ordered categories $\{c_{i_1}, \ldots, c_{i_n}\}$ is retrieved.

Finally, the continuous variable is obtained by transforming the category c_{i_j} at the j-th position in the re-ordered list into the value $(j-1)/n$. Hence, the resulting continuous variable is normalized in [0, 1] and its pyramid can be directly built and used during the simulation process.

3.4 Extension to more complex simulations

The pyramids can also be used to jointly simulate multiple variables. In this situation, several variables (properties) are defined in the TI and generated in the simulation grid. For each variable, which can be categorical or continuous, one pyramid is built and used during the simulation procedure. In particular, one can consider non-stationary TIs with auxiliary variable(s) describing the non-stationarity depicted by the main variable. The auxiliary variables are usually exhaustively given in the simulation grid, which allows to control which kind of structures are simulated depending on the location.

Moreover, classical geometrical transformation (rotation, scaling) are also compatible with the use of pyramids: the rotation angle maps and/or the scaling ratio maps are given for the simulation grid and propagated through a pyramid. Note that to build a pyramid for angles, one should use their sine and cosine in order to identify angles with a difference of 360°, because moving averages are applied.

Finally TIs containing uninformed cells can also be considered by applying reduce and expand operations over regions sufficiently informed. A simulation grid mask indicating which cells have to be simulated can also be propagated through the pyramid of the SG.

4 Results

In this section, MPS simulations are performed with different set-ups, using the algorithm DeeSse in classical mode (without pyramids) or following the proposed methodology based on pyramids. Four examples (E1–E4) are presented below to demonstrate the capability of the proposed method to handle various cases: binary variable with and without conditioning data (E1), continuous variable (E2), multi-category variable in three dimension (E3), and multi-category variable with a non-stationary TI and local geometrical transformations (E4). Pyramids are used with $L = 2$ additional levels, and a unique reduction factor for every direction and level transition of $k = 2$ and $k = 3$.

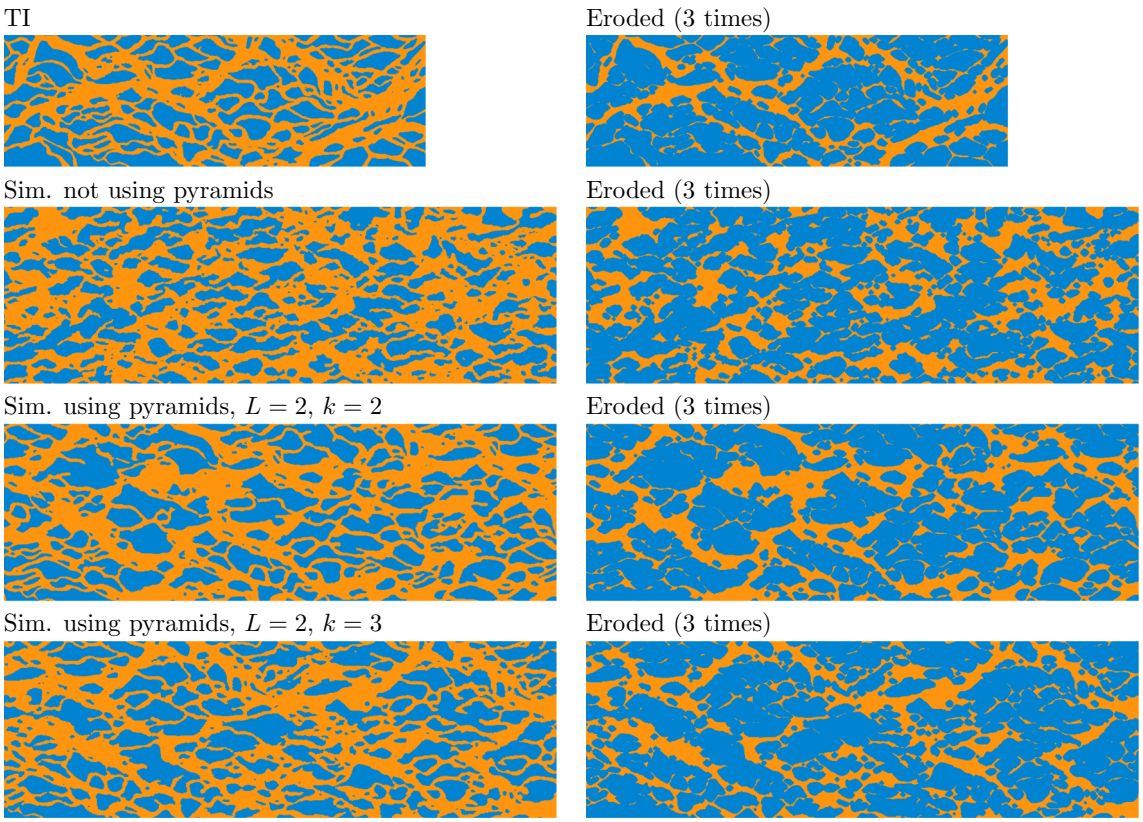

Fig. 11 Example E1-unconditional—comparison of simulations using or not pyramids. Left column) the TI (top), one unconditional realization not using pyramids (2nd row) and using pyramids with $L = 2$ and $k = 2$ (3rd row), resp. $k = 3$ (bottom). Right column) eroded images of the images at left

Moreover for multi-category variable (E3–E4), both strategies presented in Sect. 3.3 to deal with the pyramids are used. In each case, the realizations are compared with the training images, visually and also with measures such as variograms and connectivity properties. Advantages and drawbacks of the use of pyramids are discussed, and the performance in terms of computational time is given in Sect. 4.5.

4.1 Example E1: binary simulations

As a first case, the illustrative examples presented in Sect. 3.1.4 without conditioning data (E1-unconditional) and in Sect. 3.2.3 accounting for conditioning data (E1-conditional) are considered.

For the unconditional case, 50 realizations are generated: (1) not using pyramids, (2) using pyramids with $L = 2$ (number of levels additional to the initial resolution) and $k = 2$ (reduction factor for x and y direction and for every level transition), and (3) using pyramids with $L = 2$ and $k = 3$. The other parameters—$N = 24$ (maximal number of neighbors), $t = 0.02$ (acceptation threshold) and $f = 0.33$ (maximal scanned fraction)—are kept identical for the three simulation set-ups.

In Fig. 11 (left), the TI and the first realizations in each case are displayed. We observe that using pyramids helps reproduce the spatial structures at every scale. Indeed, in the TI, we can distinguish channels at two different scales: main (larger) channels and thinner ones. In the realization not using pyramids, the structures of these two scales are mixed up. On the contrary, using pyramids allows to better preserve this specific feature. This is emphasized on the eroded images (Fig. 11, right) obtained by applying three times the erosion operation onto the initial images. Such operation consists in replacing the channel facies (code 1) by the matrix facies (code 0) in every cell having at least one of its four adjacent cells (left, right, above or below) filled with the matrix facies. The eroded images show the channelized structures, while the thin channels disappear. The resulting images clearly show that the large scale structures are better reproduced with the proposed method.

The connectivity function along the x-axis corroborates this observation (Fig. 12). For a binary medium represented by an indicator function I, one defines the connectivity function

$$\tau(h) = \text{Prob}(u \leftrightarrow u + h | I(u) = I(u + h) = 1), \tag{17}$$

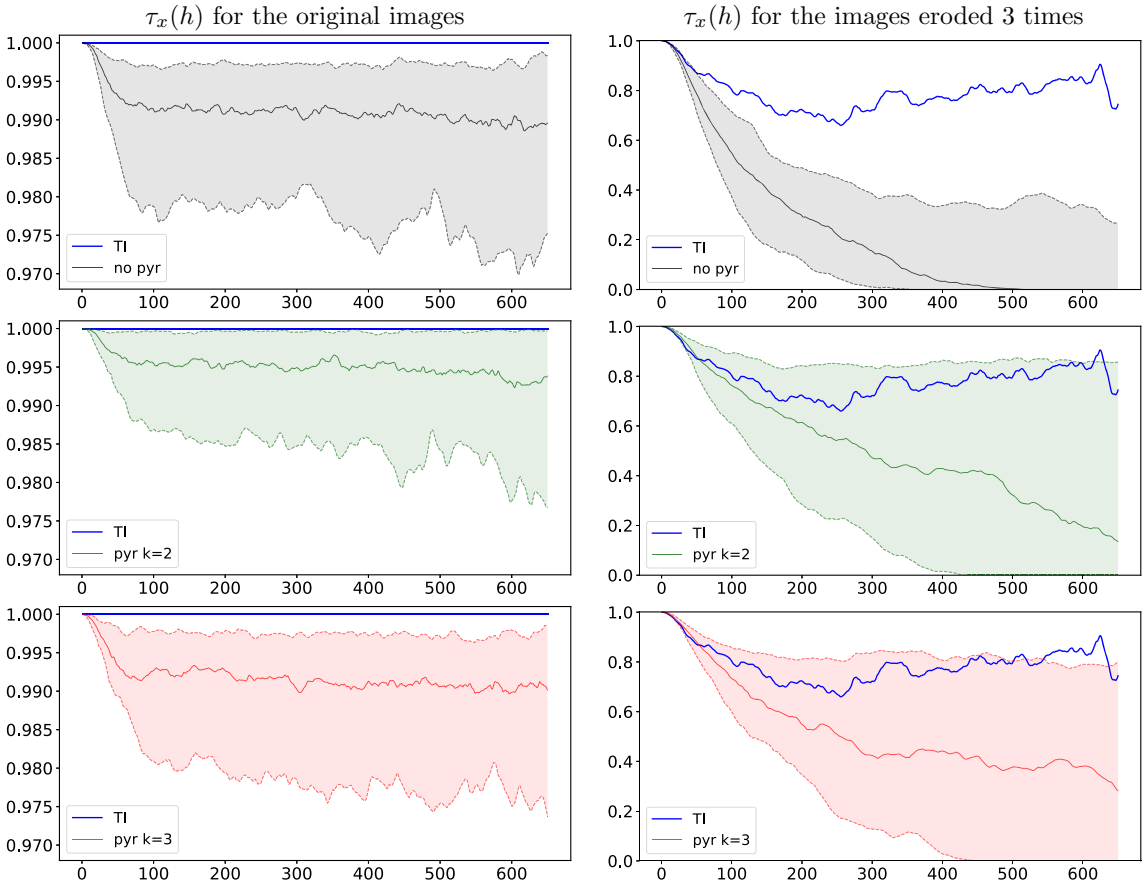

Fig. 12 Example E1-unconditional—connectivity function along x-axis for the original images (left) and the eroded images (right), for the TI (blue line) and the 50 realizations of each set-up: without pyramids (gray), using pyramids with $L = 2$, $k = 2$ (green), using pyramids with $L = 2$, $k = 3$ (red); the solid line represents the median, and the filled area the 5–95 percentile range

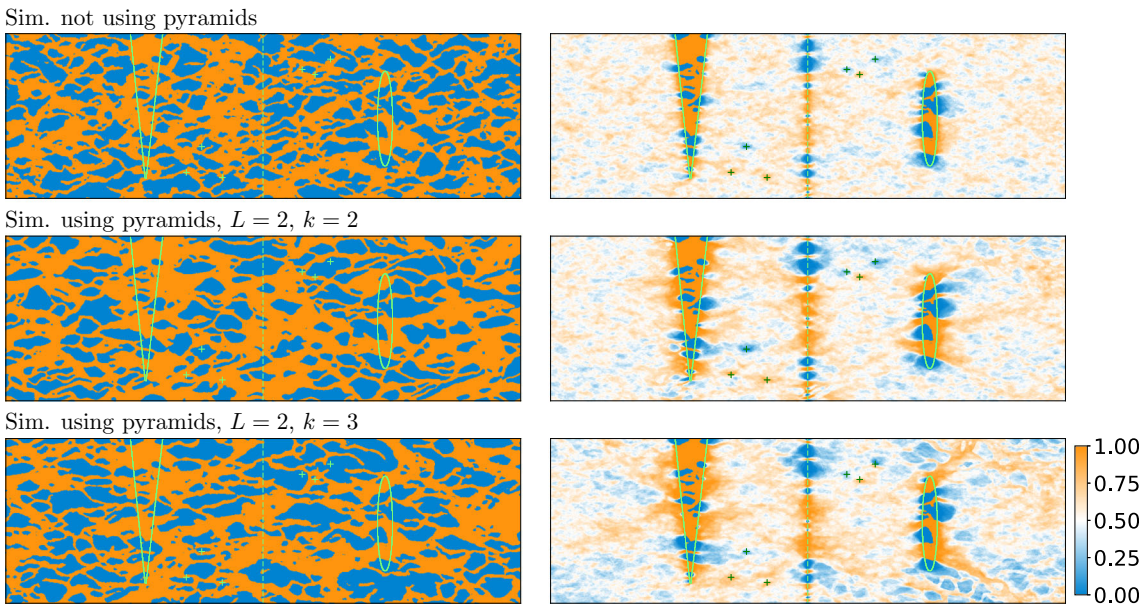

Fig. 13 Example E1-conditional—simulations using or not pyramids. Left column) one unconditional realization not using pyramids (1st row) and using pyramids with $L = 2$ and $k = 2$ (2nd row) and $k = 3$ (3rd row). Right column) pixel-wise mean over 50 realizations in each case

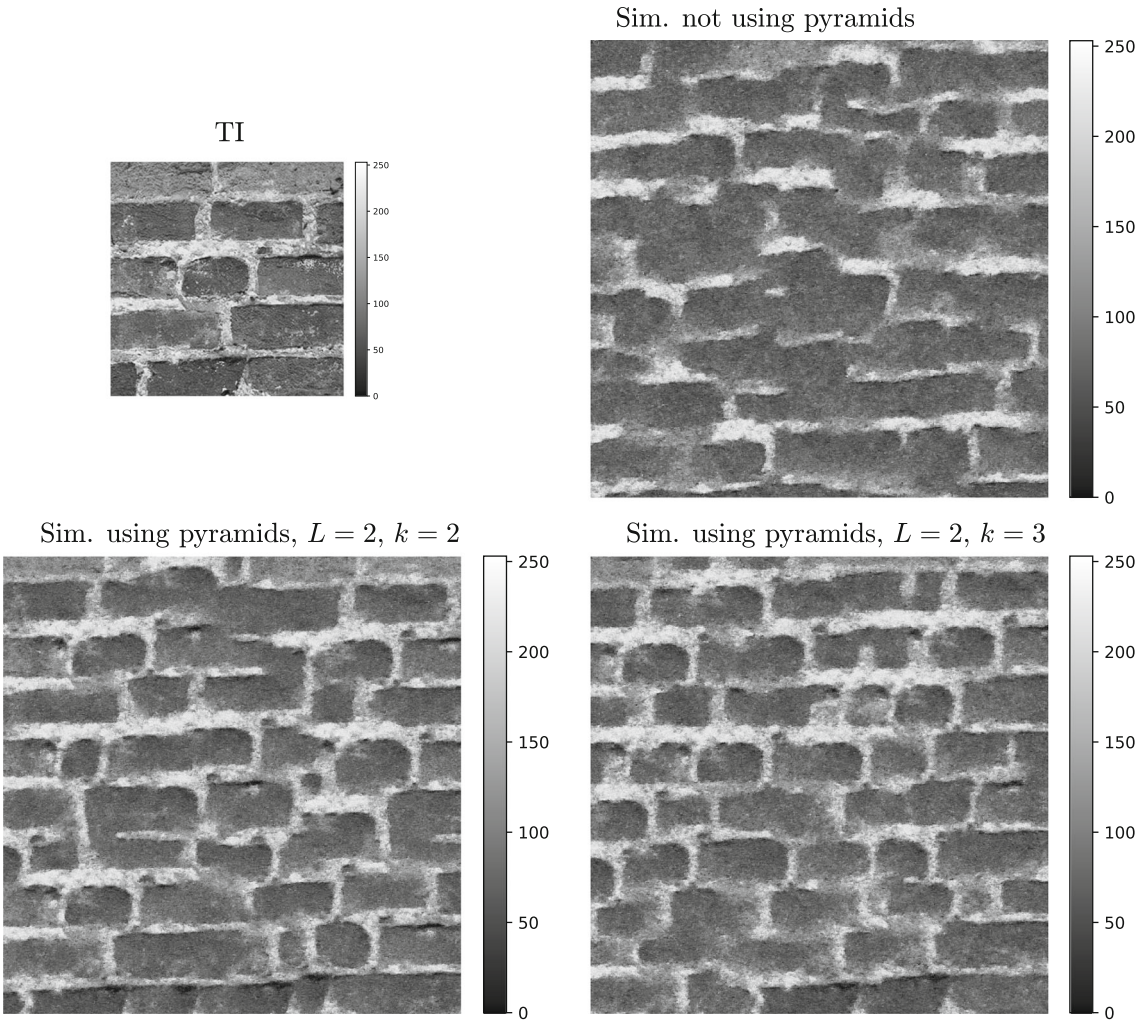

Fig. 14 Example E2—TI (256 × 256) and simulations using or not pyramids (500 × 500)

as the probability that two grid cells distant of a vector h (in number of cells) and belonging to the set of locations $I = 1(= \{u \mid I(u) = 1\})$ are connected ($u \leftrightarrow u + h$). Two cells are connected if there exists a path of adjacent cells within the set $I = 1$ linking the two nodes. Note that this function is written $\tau_x(h)$ if h is parallel to the x-axis. All channel pixels are connected in the TI, then the connectivity function is a constant equal to one (Fig. 12, left). The figure shows that using pyramids allows to better preserve the connectivity along the x-axis. Moreover, according to this measure, the reduction factor $k = 2$ performs in a slightly superior way compared to $k = 3$.

For the conditional case, the hard data set used in Sect. 3.2.3 is considered. Some results using or not pyramids are compared in Fig. 13: one conditional realization (left column) and the pixel-wise mean over 50 realizations (right column) are shown for each case. In each case, the results are consistent with the hard data points. However, the use of pyramids results in a lower variability, as we can observe on the mean maps (in the bottom right of the grid for example), with a greater impact when using the reduction factor $k = 3$.

4.2 Example E2: simulation of a continuous variable

In this example, we use a 256 × 256 continuous TI representing a texture of brick wall in grayscale (from Brooks and Dodgson (2002)). Again, 50 realizations of size 500 × 500 are generated: (1) not using pyramids, (2) using pyramids with $L = 2$, $k = 2$, and (3) $L = 2$, $k = 3$, while keeping the DeeSse parameters $N = 32$, $t = 0.02$ and $f = 0.2$ unchanged. The TI and the first realization in each case are displayed in Fig. 14.

For the TI, and the first realization of each simulation set-up, the variogram map for the lag vectors (h_x, h_y), $-200 \leqslant h_x, h_y \leqslant 200$, is computed (Fig. 15). The variance for every set-up is displayed as box-plot and compared to

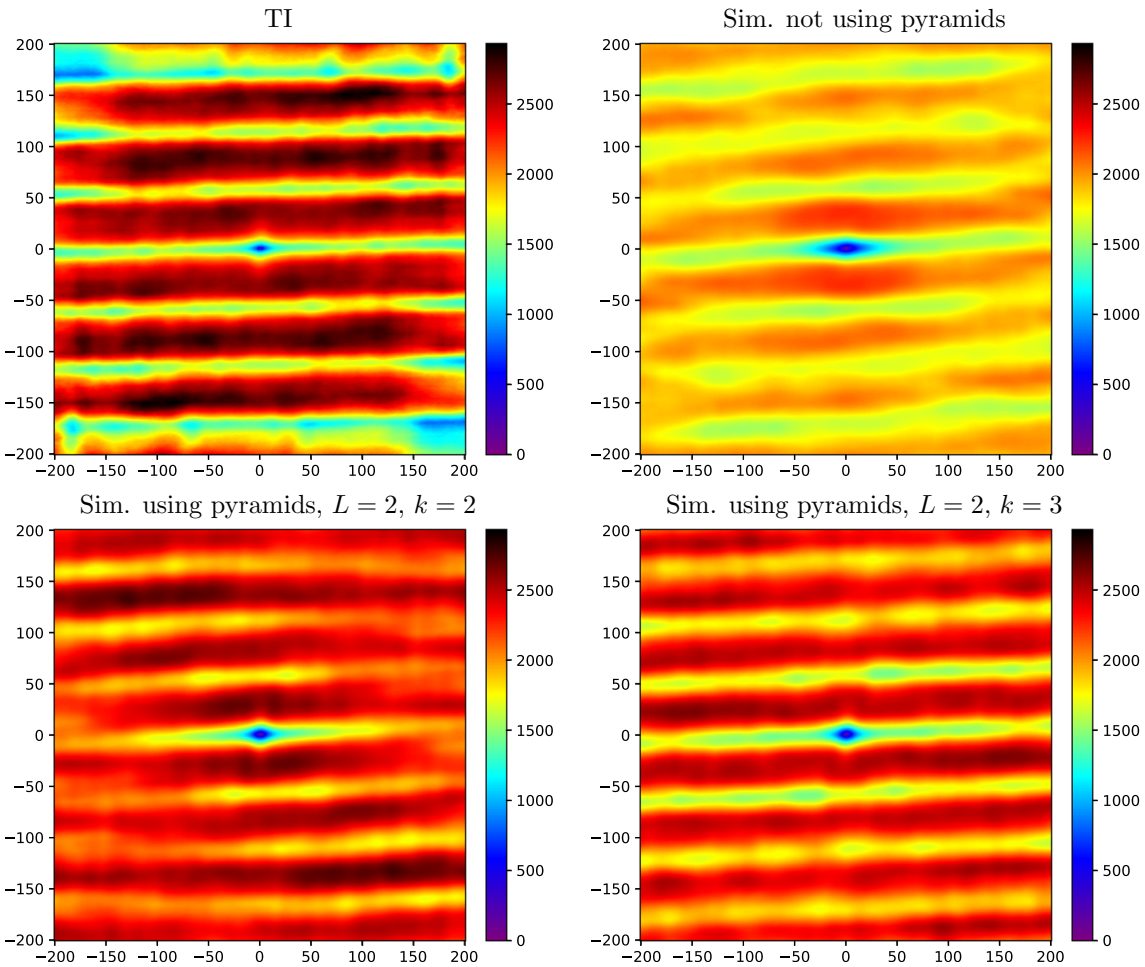

Fig. 15 Example E2—variogram maps for the TI and the first realization of each set-up

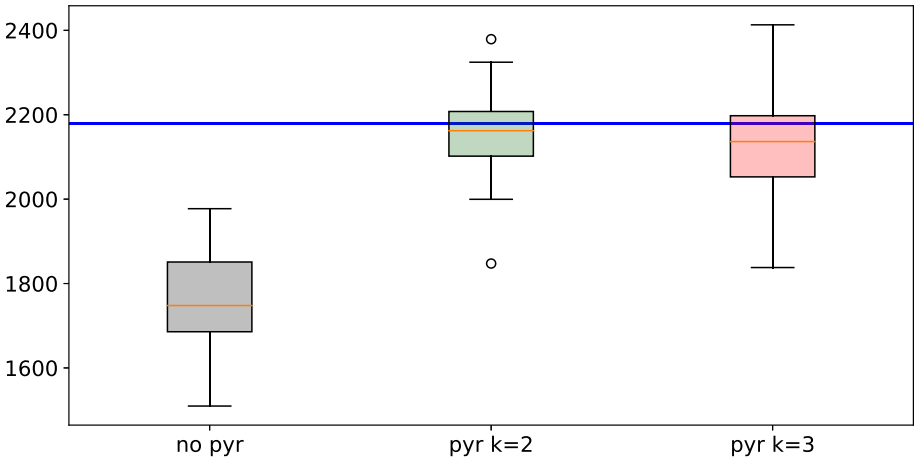

Fig. 16 Example E2—variance for the TI (blue line) and the 50 realizations of each set-up

the variance of the TI on Fig. 16. Moreover, we compute the curves Γ and Γ^c as global connectivity measures (Renard and Allard 2013), which are defined as follows. Let X be a continuous image. For a real value v, let I_v be the binary image obtained by applying a threshold to X: for any cell u, $I_v(u) = 1$ if $X(u) < v$ and $I_v(u) = 0$ otherwise. Then, the number $\Gamma(v)$ is defined as the probability that two cells

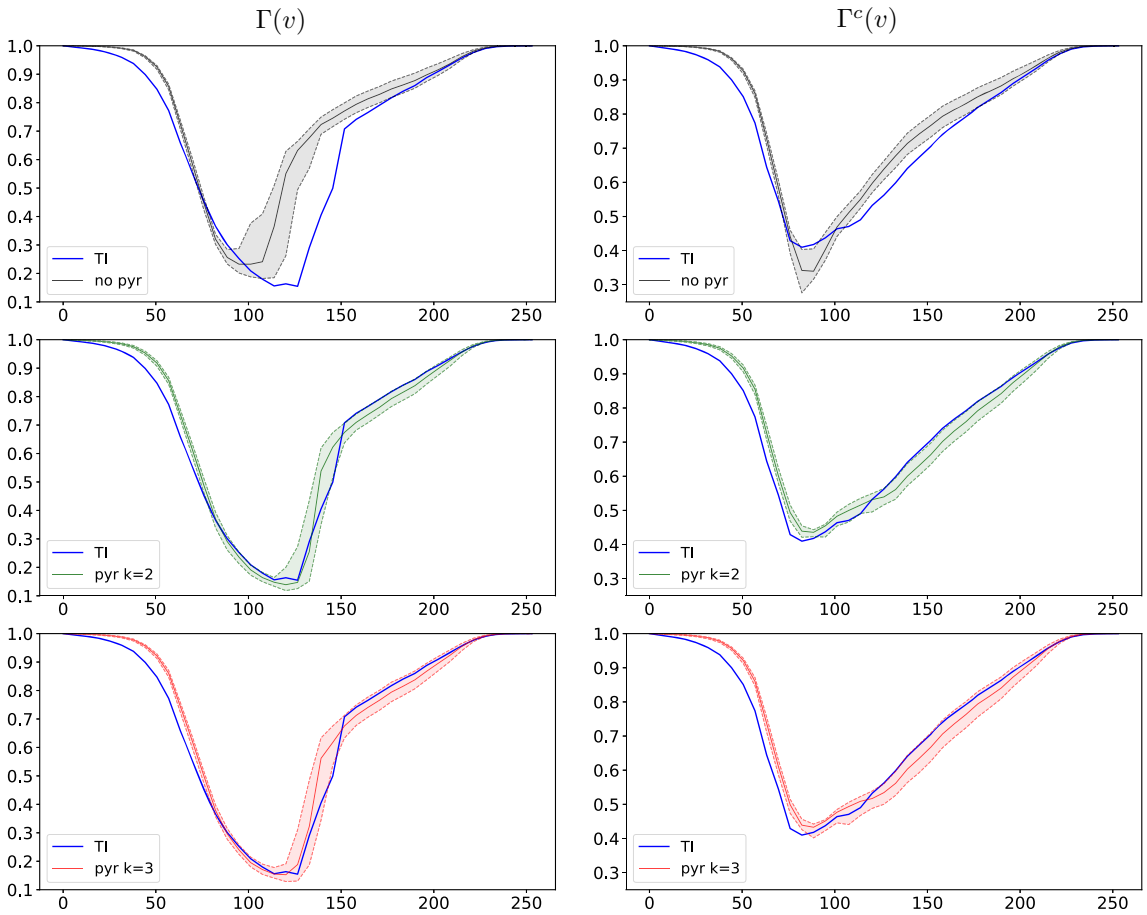

Fig. 17 Example E2—connectivity curves Γ and Γ^c, for the TI (blue line) and the 50 realizations of each set-up: without pyramids (gray), using pyramids with $L = 2$, $k = 2$ (green), using pyramids with $L = 2$, $k = 3$ (red); the solid line represents the median, and the filled area the 5–95 percentile range

randomly chosen in the set $I_v = 1$ are connected, which is computed as

$$\Gamma(v) = \frac{1}{n_p^2} \sum_{i=1}^{N(I_v)} n_i^2, \tag{18}$$

where $N(I_v)$ is the number of connected components in the set $I_v = 1$, n_p is the total number of cells in this medium, and n_i is the number of cells in its i-th connected component. The number $\Gamma^c(v)$ is defined in a similar way from the complementary image I_v^c, defined as $I_v^c(u) = 1 - I_v(u)$ for any cell u. The two curves Γ and Γ^c are obtained by taking various threshold values between the minimum and the maximum values in X. These curves are displayed for the TI and each simulation set-up in Fig. 17 (bottom row).

We observe for this example that enabling pyramids helps reproduce the structures of the TI. The variogram map of the TI is better reproduced when pyramids are used. This is true in particular for short ranges (center of the maps) and especially in the horizontal direction. Moreover, the variance of the TI is underestimated in the standard simulations, whereas the use of pyramids allows for a good estimation. Finally, the Γ- and Γ^c-curves show again the advantage of using pyramids, which allows a better reproduction of the TI characteristics. In this case and based on these statistics computed over 50 realizations, the reduction factors $k = 2$ and $k = 3$ imply similar results.

4.3 Example E3: three-dimensional simulation of a categorical variable

A 4-category three-dimensional TI of size $100 \times 94 \times 60$ representing channels with levees (Fig. 18, top left) is used and 50 realizations of dimensions $75 \times 150 \times 50$ are generated for each of the five simulation set-ups: (1) not using pyramids, (2a) using pyramids with $L = 2$, $k = 2$ and based on the indicator variables of categories (Sect. 3.3.1), (2b)

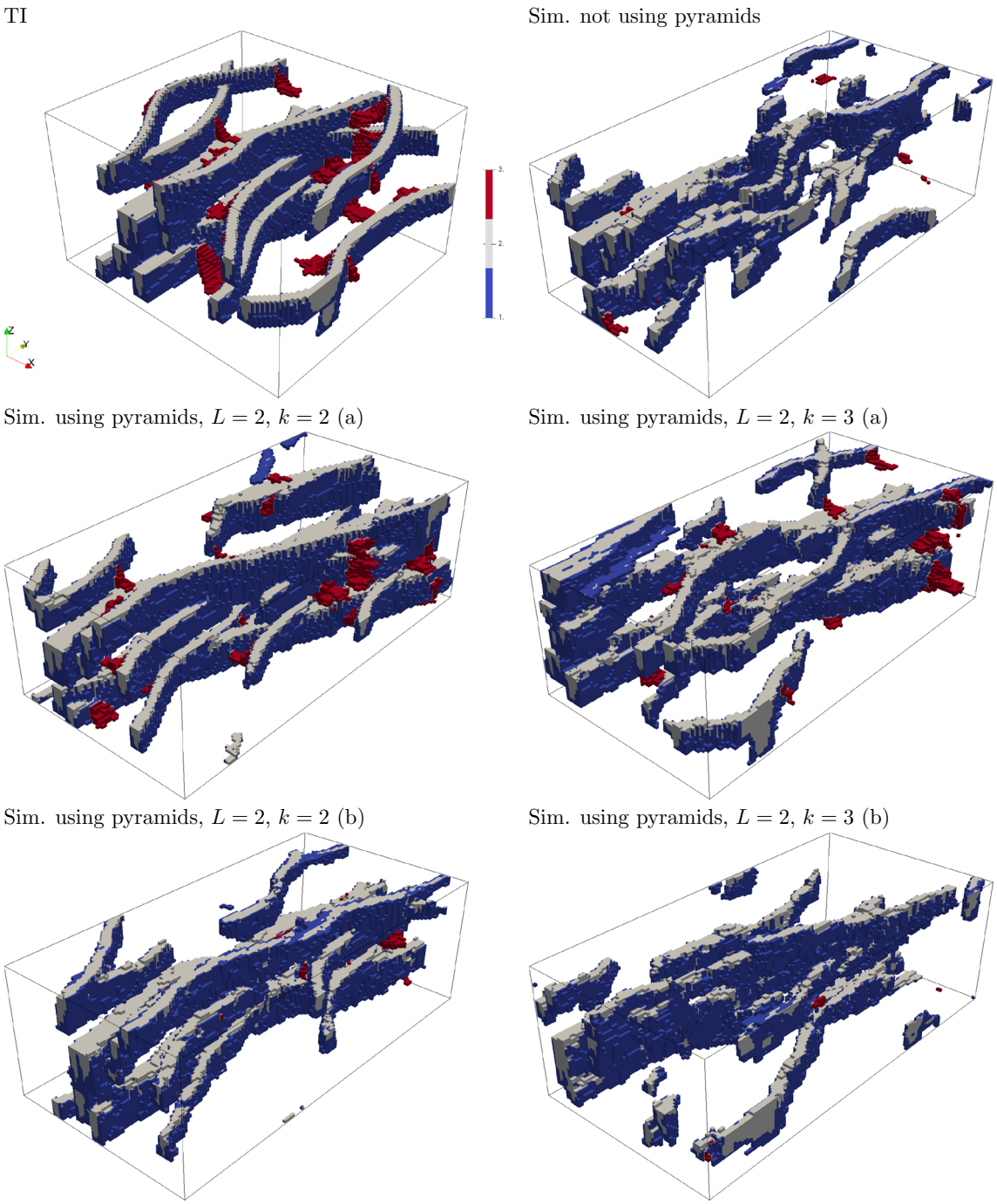

Fig. 18 Example E3—4-category TI ($100 \times 94 \times 60$) (courtesy of Total S.A.), and simulations using or not pyramids ($75 \times 150 \times 50$); **a** pyramids based on the indicator variables of categories, **b** pyramids based on a representative continuous variable

using pyramids with $L = 2$, $k = 2$ and based on a representative continuous variable (Sect. 3.3.2), and (3a, 3b) same set-ups as the two previous ones but with $k = 3$. The DeeSse parameters are set to $N = 32$, $t = 0.05$ and $f = 0.3$ for each case. One realization for each case is displayed in Fig. 18. Visually, the simulations appear a little bit "smoother" with the reduction factor $k = 2$ compared to the results with $k = 3$.

The proportion of each facies code is computed for the TI and each realization, and shown in Fig. 19. We observe that the proportion of facies codes are better reproduced when pyramids are used with the technique (a). The

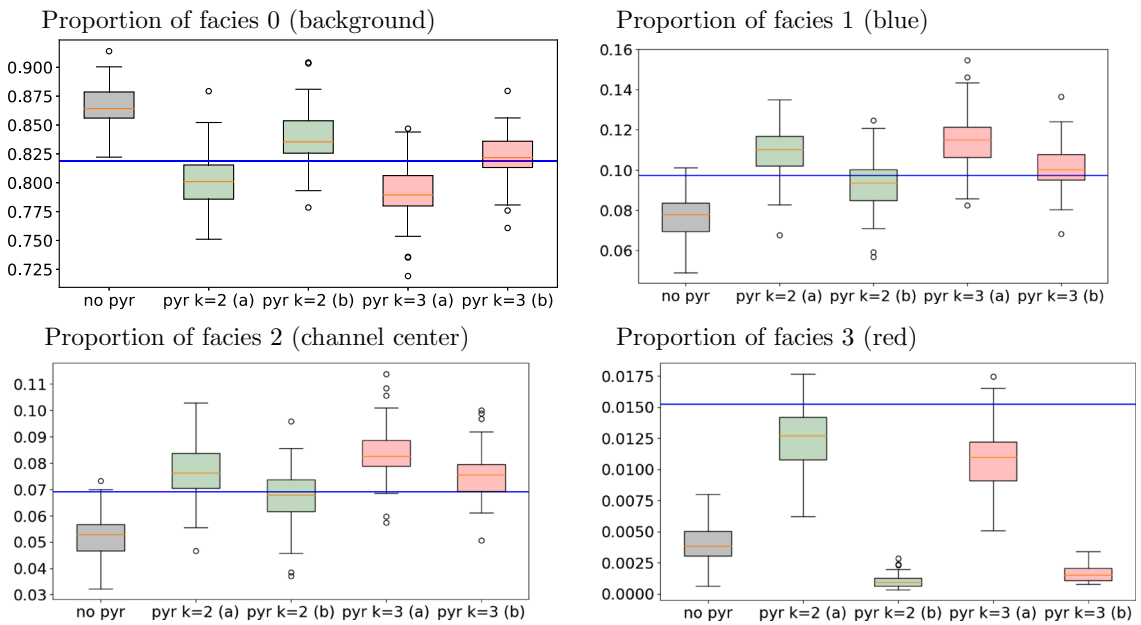

Fig. 19 Example E3—facies proportion for the 50 realizations of each set-up; **a** pyramids based on the indicator variables of categories, **b** pyramids based on a representative continuous variable; the horizontal line corresponds to the proportion in the TI

technique (b) fails to reproduce the facies code 3, which has the smallest proportion in the TI and becomes almost absent in the simulations.

The facies code 2 is the facies in the center of the channels whose main direction is the y axis. Then, we check the quality of the simulations with respect to the connectivity function $\tau_y(h)$ (Eq. 17) along this axis and for the indicator variable of this facies. These curves are displayed in Fig. 20, and we observe that the connectivity along the y-axis is much longer when using pyramids. In particular, most of the realizations without pyramids do not contain any traversing channels from one border to the other. The use of pyramids allows to remediate to this problem.

4.4 Example E4: complex simulation

This example shows that using pyramids allows for complex simulation set-ups. We consider a 1000×700 bivariate TI composed of a categorical variable (4 facies) and an auxiliary continuous variable describing the non-stationarity present in the facies image consisting in a horizontal trend according to the facies code 0 (blue) and 1 (light blue) (Fig. 21). We propose to generate realizations of the categorical variable in a simulation grid of size 700×700, where some cells in the borders are masked (not simulated), with rotations of the structures present in the TI according to the map of Fig. 22 (left), and by controlling the features of the blue (background) facies code 0 and 1 by giving the auxiliary continuous variable of Fig. 22 (right). The gray cells in the maps of Fig. 22 correspond to masked cells.

In this situation, bi-variate simulations are done, where the continuous variable is exhaustively known (every cell is indeed a conditioning cell). Simulation using pyramids involves a pyramid for the angle map, the mask map, and both variables. For the categorical variable, one can use either the technique based on the indicator variables of categories (a) or based on a representative continuous variable (b). One realization is displayed in Fig. 23 for the following cases: (1) not using pyramids, (2a, 2b) using pyramids with $L = 2$, $k = 2$, and (3a, 3b) $L = 2$, $k = 3$. The DeeSse parameters are set to $N = 8, 32$ for the auxiliary variable and the categorical variable respectively, $t = 0.02$ for both variables and $f = 0.3$ for each case.

Although subjective, based on a visual inspection, the set-up (2a), i.e. simulation using pyramids with $L = 2$, $k = 2$ and based on the indicator variables of categories, gives the realization of the best quality: channels are smoothly reproduced, less broken than with other set-ups. We also observe that the technique based on a representative continuous variables to deal with the pyramid of the

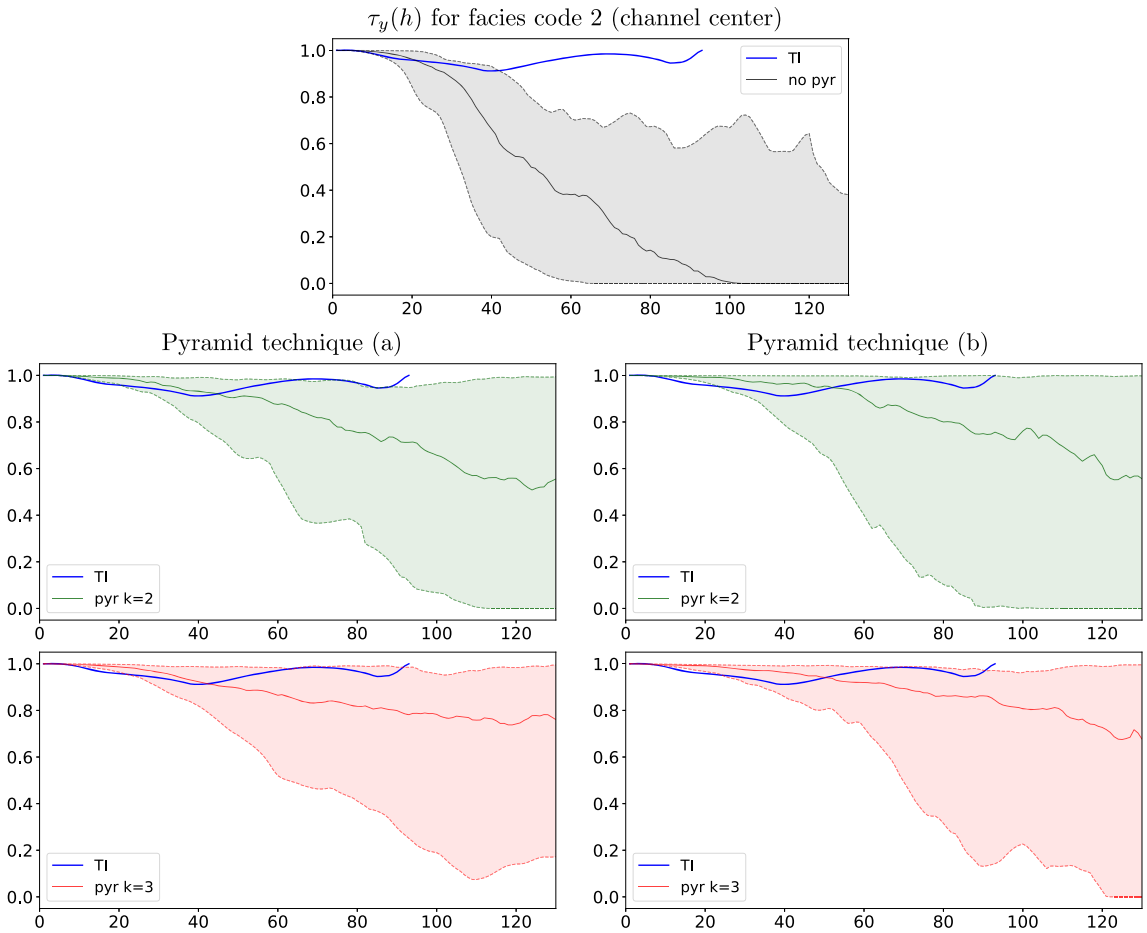

Fig. 20 Example E3—connectivity function along y-axis for the indicator variable of the facies code 2, for the TI (blue line) and the 50 realizations of each set-up: without pyramids (gray), using pyramids with $L = 2$, $k = 2$ (green), using pyramids with $L = 2$, $k = 3$ (red); the solid line represents the median, and the filled area the 5–95 percentile range; left column) pyramids based on the indicator variables of categories, right column) pyramids based on a representative continuous variable

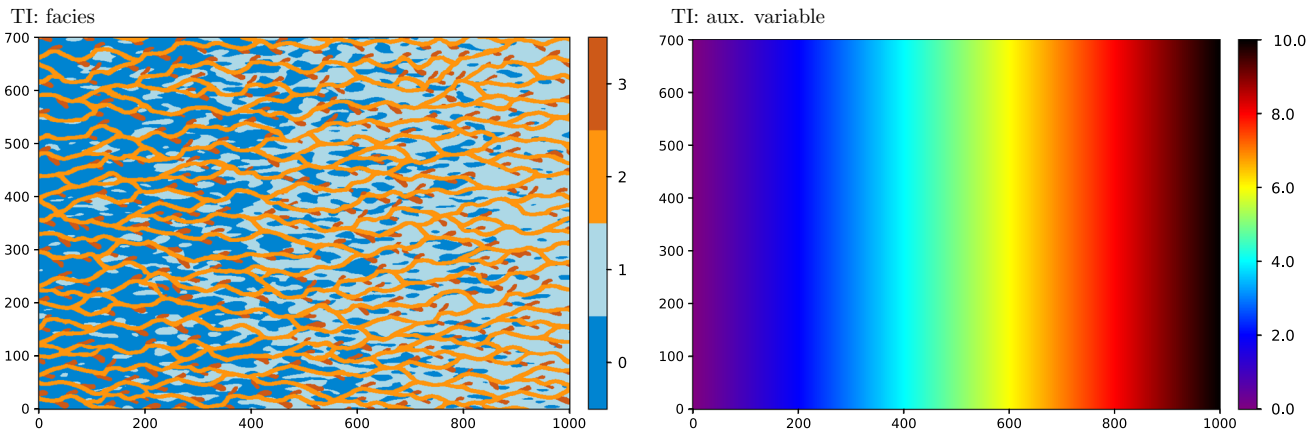

Fig. 21 Example E4—bi-variate TI (1000×700)

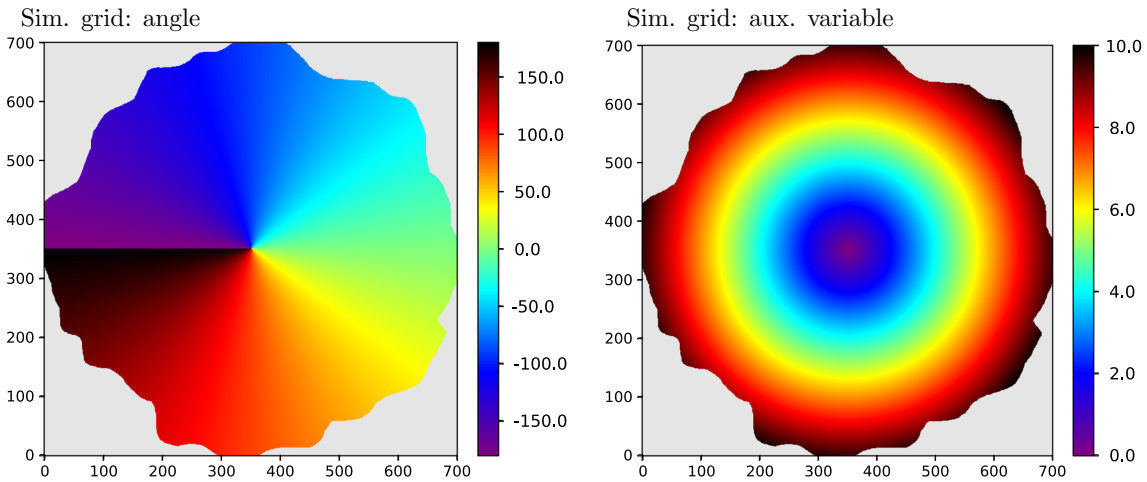

Fig. 22 Example E4—input for simulation grid (700×700)

categorical variable (cases 2b, 3b) does not allow for a proper reproduction of the spatial features given in the TI: the channels are much more broken than with the other technique, and as for the previous example, the less frequent facies in the TI tends to disappear in the simulations.

4.5 Computational performance

Table 1 shows the real elapsed time in seconds for each of the previous examples and for each simulation set-up. All tests were conducted in parallel with 8 CPUs of type *Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz*.

Note that the examples E1-uncond., E1-cond., E2 and E3 were runned with 50 realizations, the example E4 with 10 realizations, and the computing times presented in the table are given for one realization, obtained by simply dividing the whole elapsed times of the runs by the number of generated realizations. Every run comprises some pre-processing computations (done only once) consisting essentially in normalizing the TI and building the pyramid of the TI (when this option is used), see the left part of flowchart in Fig. 6. The computing time of the pre-processing part had been estimated in each case by simply running the program with zero realization (but doing the pre-processing steps) on the same type of machine with also 8 CPUs, and retrieving the elapsed time: less than 1s for the three set-ups of examples E1-uncond., E1-cond., and E2, also less than 1s for the five set-ups of example E3, and about 1s for the five set-ups of example E4. This is quite fast because the reduce and expand operations performed for building the pyramid are fully parallelized. Thus, it is important to notice that the elapsed times displayed in Table 1 can be "fairly" compared, since the time required by the pre-processing computations is negligible.

For the three first examples (E1-uncond., E1-cond., E2), the runs using pyramids are faster, with a larger gain with the reduction factor $k = 3$. For the categorical cases (examples E3 and E4), the technique using the pyramids based on the indicator variable of categories (a) is slower than the standard DeeSse without pyramids, but the technique based on a representative continuous variable (b) is faster.

Guiding the simulation with a lower resolution representation allows reducing the number of neighbors (in searched patterns) during the simulation and accelerating the algorithm. A noticeable exception is the multi-categorical case (a), because the number of simulated variables in the pyramid levels is larger. Using the re-numbering technique (b) accelerates the algorithm at the cost of a loss of quality in the results.

In summary, according to the observations made for each example in the previous sections, enabling pyramids is almost always advantageous because it allows most often for the generation of realizations of better quality in less time.

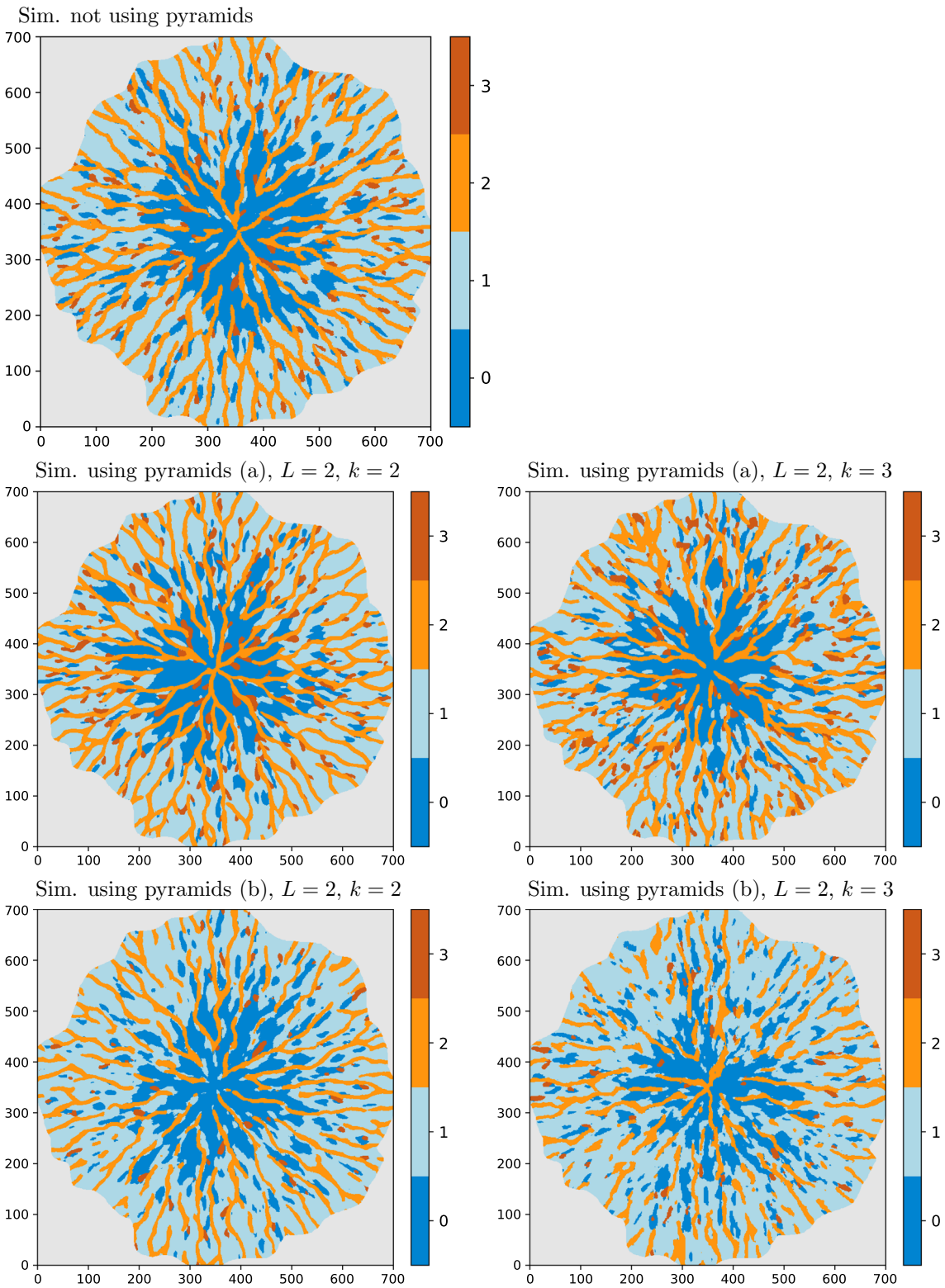

Fig. 23 Example E4—simulations using or not pyramids (700×700)

Table 1 Real elapsed time in seconds for one realization (mean over the set of realizations), for each example and each set-up

Example	No pyr.	Pyr. $k = 2$		Pyr. $k = 3$	
E1-uncond.	39	25 (64%)		16 (41%)	
E1-cond.	38	26 (70%)		17 (44%)	
E2 (cont.)	91	58 (64%)		54 (59%)	
		(a)	(b)	(a)	(b)
E3 (3D categ.)	131	142 (108%)	85 (65%)	210 (160%)	116 (88%)
E4 (complex)	476	702 (148%)	263 (55%)	549 (115%)	229 (48%)

Columns (a) refer to the technique based on the indicator variables of categories, and columns (b) to the technique based on a representative continuous variable. Proportion with respect to the time without pyramids is given in percent

5 Conclusions

In this paper a new methodology for multiple-point statistics (MPS) simulation is proposed. It consists in (1) building a pyramid for the training image (TI), i.e. successive representations at lower resolutions of the original image by computing moving averages based on a Gaussian-like kernel (reduce operation), and (2) performing successive MPS simulations in each pyramid level, from the lowest resolution to the original (finest) resolution, while the link between two levels is obtained by expanding the result of the simulation to the finer level (expand operation, pseudo-inverse of the reduce operation), the expanded image serving as conditioning variable. The direct sampling algorithm DeeSse is used as it handles joint simulations. The categorical variables can be addressed in two ways, considering one pyramid for the indicator variable for each category except one, or with one pyramid for a representative normalized continuous variable built with respect to the most frequent contacts between the categories. The latter approach guides the simulation more approximately but is faster.

The examples show that multi-resolution images (pyramids) allow to obtain simulations of better quality. The proposed method helps catch the structures at different scales. Most often the spatial features of the TI are better reproduced in terms of connectivity properties. The main drawback of the use of pyramids is that it results in a lower variability in the realizations. Moreover, using a larger reduction factor—or equivalently a wider kernel—for the pyramids, which involves more blurred images at coarser resolutions, speeds the algorithm up but tends to deteriorate the quality of the results. Hence, a reduction factor of $k = 2$ is recommended for the level transitions. Note that the number L of coarse levels in the pyramids should be chosen such that the dimensions (in number of cells) of the lowest resolution grid become not too small, providing enough patterns repetitions and making the MPS simulations reliable.

In terms of computational time the use of pyramids is also beneficial in most of the situations compared to the standard simulation without pyramids, except for the multi-categorical case based the indicator of the categories.

The wide flexibility of the DeeSse algorithm is kept with the implementation of the pyramids. It handles categorical and continuous variables, multi-variate simulation, non-stationarity, and geometrical transformation given by scaling or rotation of the TI features.

Finally, it is important to notice for the user that the integration of the pyramids approach in the DeeSse algorithm does not affect its usability. Indeed, only the number of pyramid levels and the size of the Gaussian filter (reducing factors) are additionally required as input, the simulation parameters—number of neighbors, acceptation threshold, maximal scan fraction—involved in the simulation at each pyramid level being automatically computed.

Acknowledgements We are grateful to Loïc Mercerat who carried out a preliminary study helpful for this research. This work was supported partly by the Swiss National Science Foundation (*Phenix* project, grant number 200020_182600). We also thank Total S.A. for co-funding this work.

References

Arpat G, Caers J (2007) Conditional simulation with patterns. Math Geol 39(2):177–203. https://doi.org/10.1007/s11004-006-9075-3

Brooks S, Dodgson N (2002) Self-similarity based texture editing. ACM Trans Graph 21(3):653–656. https://doi.org/10.1145/566654.566632

Burt P, Adelson E (1983) The laplacian pyramid as a compact image code. IEEE Trans Commun 31(4):532–540. https://doi.org/10.1109/TCOM.1983.1095851

Gardet C, Le Ravalec M, Gloaguen E (2016) Pattern-based conditional simulation with a raster path: a few techniques to make it more efficient. Stoch Environ Res Risk Assess 30(2):429–446. https://doi.org/10.1007/s00477-015-1207-1

Mahmud K, Mariethoz G, Caers J, Tahmasebi P, Baker A (2014) Simulation of Earth textures by conditional image quilting. Water Resour Res 50(4):3088–3107. https://doi.org/10.1002/2013WR015069

Mariethoz G, Renard P, Straubhaar J (2010) The Direct Sampling method to perform multiple-point geostatistical simulations. Water Resour Res 46:1–14. https://doi.org/10.1029/2008WR007621

Oriani F, Straubhaar J, Renard P, Mariethoz G (2014) Simulation of rainfall time series from different climatic regions using the direct sampling technique. Hydrol Earth Syst Sci 18(8):3015–3031. https://doi.org/10.5194/hess-18-3015-2014

Renard P, Allard D (2013) Connectivity metrics for subsurface flow and transport. Adv Water Resour 51:168–196. https://doi.org/10.1016/j.advwatres.2011.12.001 **(35th Year Anniversary Issue)**

Rezaee H, Mariethoz G, Koneshloo M, Asghari O (2013) Multiple-point geostatistical simulation using the bunch-pasting direct sampling method. Comput Geosci 54:293–308. https://doi.org/10.1016/j.cageo.2013.01.020

Shahraeeni M (2019) Enhanced multiple-point statistical simulation with backtracking, forward checking and conflict-directed back-jumping. Math Geosci 51(2):155–186. https://doi.org/10.1007/s11004-018-9761-y

Straubhaar J (2019) DeeSse user's guide. The Centre for Hydrogeology and Geothermics (CHYN), University of Neuchâtel, Neuchâtel, Switzerland

Straubhaar J, Renard P, Mariethoz G, Froidevaux R, Besson O (2011) An improved parallel multiple-point algorithm using a list approach. Math Geosci 43(3):305–328. https://doi.org/10.1007/s11004-011-9328-7

Straubhaar J, Walgenwitz A, Renard P (2013) Parallel multiple-point statistics algorithm based on list and tree structures. Math Geosci 45(2):131–147. https://doi.org/10.1007/s11004-012-9437-y

Straubhaar J, Renard P, Mariethoz G, Chugunova T, Biver P (2019) Fast and interactive editing tools for spatial models. Math Geosci 51(1):109–125. https://doi.org/10.1007/s11004-018-9766-6

Strebelle S (2002) Conditional simulation of complex geological structures using multiple-point statistics. Math Geol 34(1):1–21. https://doi.org/10.1023/A:1014009426274

Strebelle S, Remy N (2005) Post-processing of multiple-point geostatistical models to improve reproduction of training patterns. In: Leuangthong O, Deutsch C (eds) Geostatistics Banff 2004. Springer, Berlin, pp 979–988

Suzuki S, Strebelle S (2007) Real-time post-processing method to enhance multiple-point statistics simulation. In: Petroleum geostatistics. EAGE, Cascais, Portugal

Tahmasebi P, Hezarkhani A, Sahimi M (2012) Multiple-point geostatistical modeling based on the cross-correlation functions. Comput Geosci 16(3):779–797. https://doi.org/10.1007/s10596-012-9287-1

Zhang T, Switzer P, Journel A (2006) Filter-based classification of training image patterns for spatial simulation. Math Geol 38(1):63–80. https://doi.org/10.1007/s11004-005-9004-x