Non-stationary Multiple-point Geostatistical Simulations with Region Concept

Jianbing Wu\(^{(1)}\), Tuanfeng Zhang\(^{(2)}\) and Alexandre Boucher \(^{(1)}\)

\(^{(1)}\) Stanford Center for Reservoir Forecasting, Stanford, CA
\(^{(2)}\) Schlumberger Doll Research, Boston, MA

May, 2007

Abstract

The multiple-point (mp) geostatistical algorithm SNESIM can generate ‘realistic’ lithofacies model by borrowing geological patterns from a training image. However, it requires that input training image to be reasonably stationary. In practice, most reservoirs are non-stationary, thus the incorporation of non stationary information during mp simulation is a challenge.

In this paper, two concepts are proposed to account for non stationary constraints in the SNESIM algorithm: (1) extended rotation/affinity concept and (2) region concept. Local rotation/affinity can handle local non stationarity whenever the geological structures differ only by orientations and sizes. The implementation is extremely CPU efficient, at the cost of greater RAM demand because of the large number of search trees that must be constructed and stored in memory. The region concept is designed to account for the ‘general’ non stationary case when the geological patterns are deemed significantly different from one reservoir sub-domain (region) to another, calling for different training images and parameters for each individual region. Implementation of this region concept is both RAM and CPU efficient, while preserving simulation quality. This region concept is also implemented into the FILTERSIM algorithm to account for the general non stationary case.
1 Introduction

Prior to the introduction of multiple-point Geostatistics, two large families of simulation algorithms for facies modeling were available: pixel-based (Matheron, 1970; Deutsch and Journel, 1998) and object-based (Leeder, 1978; Mackey and Bridge, 1992). The pixel-based algorithms build the simulated realizations one pixel at a time, thus providing great flexibility for conditioning to data of diverse support volumes and diverse types. However, pixel-based algorithms have difficulty reproducing complex geometric shapes, particularly if simulation of these pixel values is constrained only by two-point statistics, such as a variogram or a covariance. Object-based algorithms build the realizations by dropping onto the simulation grid one object or pattern at a time; hence they can be fast and faithful to the geometry of the object. However, they are difficult to condition to local data of different support volumes, particularly when these data are dense as in the case of seismic surveys.

The multiple-point simulation (mps) concept proposed by Journel (1992) and first implemented by Guardiano and Srivastava (1992), combines the strengths of the previous two classes of simulation algorithms. It operates pixel-wise with the conditional probabilities for each pixel value being lifted as conditional proportions from a training image depicting the geometry and distribution of objects deemed to prevail in the actual field. The mps concept became practical with the SNESIM (Single Normal Equation SIMulation) implementation of Strebelle (2000). In SNESIM, the training image is scanned only once; all conditional proportions available in that training image for a given search template size are faithfully stored in a smart search tree data structure, from which they can be retrieved fast. The SNESIM algorithm contains two main parts, (1) the construction of the search tree where all training proportions are faithfully stored, and (2) the simulation part where these proportions are read and used to draw the simulated values.

SNESIM was designed for simulations of categorical variables, e.g. facies distributions. SNESIM can be memory-demanding, especially when the training image is large with a large number of categories and a rich variety of different patterns. Hence SNESIM can only handle a limited number of categories ($\leq 4$), and it does not work for continuous variables. The characteristic of the SNESIM algorithm is that it strives to be faithful to the conditioning data event by not accepting any training replicate that does not fully identify the conditioning data values.
The **FILTER**-based **SIMulation** mp algorithm **FILTERSIM** \cite{Zhang2006} was introduced to circumvent these problems. The **FILTERSIM** algorithm is much less memory demanding yet with a reasonable CPU cost, and can handle both categorical and continuous variables. This is obtained by accepting approximate replicates of the conditioning data event. **FILTERSIM** utilizes a few linear filters to classify training patterns in a filter space of reduced dimension. Similar training patterns are grouped and stored in a class whose prototype pattern is the point wise average of all patterns falling into that class. During simulation, the prototype closest to the conditioning data event is determined. A training pattern from that prototype class is then drawn, and pasted back onto the simulation grid. The process is similar to constructing a jigsaw puzzle drawing from stacks of similar-looking pieces.

Instead of saving faithfully all training replicates in a search tree as does **SNESIM**, **FILTERSIM** only saves the central location of each training pattern in memory, hence reducing RAM demand.

## 2 Motivations

The prerequisite of the **SNESIM** and **FILTERSIM** algorithms is availability of a training image reasonably stationary so that meaningful statistics can be inferred by scanning it. In a stationary training image, the data patterns or structures should repetitively appear over the entire image, instead of being location-specific in that image. As any other spatial statistics, the conditional proportions called by a mp algorithm, such as **SNESIM**, can only capture stationary information (image patterns or structures). During the simulation process, those stationary patterns are pasted back onto the simulation grid after proper anchoring to the local conditional data events.

However, most reservoirs are non-stationary, with different sub-domains featuring different geological structures. Figure 1(a) shows a fluvial deltaic fan reservoir, which is non-stationary: the thick channels are in the top left corner and channels get thinner towards the bottom right corner; also the channel orientation gradually changes from quasi horizontal in the top to vertical in the bottom. The important practical issue is to adapt **SNESIM** so that it can handle such non-stationary information, more generally statistics and patterns that vary in space.

A non stationary reservoir cannot be simulated directly with an equally non stationary training image. Performing **SNESIM** with the non stationary training image shown in Figure 1(a) will generate non-conditional realizations of the type...
Figure 1: *SNESIM* simulation with a non-stationary training image (black: channel; gray: mudstone), both images are of same size.

shown in Figure 1(b). The realization of Figure 1(b) displays stationary (non location-specific) patches of training image patterns of dimension equal to that of the data search neighborhood used. Location-specific, non-stationary, patterns can NOT be reproduced directly by a single run of *SNESIM*.

Zhang (2002) adapted for mps the concept of local rotation and affinity (Journel and Xu, 1994), initially developed for non stationary variogram models. In Zhang’s implementation (2002), the non-stationary local data event (dev) is extracted first with the predefined search template; it is then corrected into $dev^*$ by rotation and affinity to match the stationary statistics of the training image; this transformed $dev^*$ is used to retrieve the conditional probability from the pre-built search tree, see Figure 2. This implementation of rotation/affinity concept calls for a CPU-intense transformation of the local data event at each simulated location. Moreover, this method only allows using one single stationary training image. Hence Zhang’s implementation is impractical when the reservoir features significantly different geological scenarios from one sub-domain to another. A more general, yet CPU-efficient, algorithm is needed to account for such local non-stationary constraints.

Two approaches are proposed to handle non-stationary simulation: (1) modify locally the training image by rotation/rescaling; (2) use different training images through a region concept. The first approach is presented in Section 3 and im-


Figure 2: Data event manipulation (taken from [Caers and Zhang (2004)])

plemented in the SNESIM algorithm; the second approach is detailed in Sections 4 and 5 and it is implemented in both SNESIM and FILTERSIM algorithms.

3 Extended Rotation/Affinity Concept

Given one single input training image, the rotation/affinity is applied to transform locally the original input search template into a new search template which is used to retrieve the training conditional probability. A set of different search trees is built to accommodate the different transformed search templates. This is equivalent to keeping the search template untouched but rotate and rescale the original training image into a set of new training images. Our implementation is CPU-efficient at the cost of greater RAM demand, since many search trees must be built and simultaneously stored. Zhang’s implementation (2002) was CPU-intensive but calls only for one search tree.

3.1 Rotation and scaling regions

The simulation field \( G \) can be divided into \( N_{\text{rot}} \) rotation regions, each region \( R_i \) (\( i = 1, \cdots, N_{\text{rot}} \)) associated with a specific rotation angle \( r^i \), where \( r^i \) is the
azimuth rotation angle about the (vertical) Z-axis in the \(i^{th}\) region \(R^i\). In each rotation region \(R^i\), the original training image \(TI\) is rotated \(r^i\) degrees into a new training image \(TI_i\), and the search tree of that new training image is used for \textit{SNESIM} simulation only in region \(R^i\). In our implementation of \textit{SNESIM} one single azimuth rotation is allowed around the Z-axis, with the angle measured in degree increasing clockwise from the Y-axis.

Next, the simulation grid \(G\) is divided into a set of scaling (affinity) regions, each region being associated with different scaling factors in the three X/Y/Z directions. For each scaling region \(S^j\) \((j = 1, \ldots, N_{aff})\), its scaling factors (also called affinity ratios) are \(f^j = \{f^j_x, f^j_y, f^j_z\}\), where \(N_{aff}\) is number of regions for scaling, \(S^1 \cup \cdots \cup S^{N_{aff}} = G\), and \(f^j_x, f^j_y, f^j_z\) are the affinity factors in the three X/Y/Z directions, respectively. Note that all affinity factors must be positive \(\in (0, +\infty)\). An affinity factor equal to 1 means no training image rescaling in that direction. In the \(j^{th}\) scaling region \(S^j\), the original training image \(TI\) of size \(N_x \times N_y \times N_z\) is transformed into a new training image \(TI_j\) of size \((N_x \cdot f^j_x) \times (N_y \cdot f^j_y) \times (N_z \cdot f^j_z)\). The larger the affinity factor, the larger/thicker the geological structure in that direction. The search tree of that new training image is used for \textit{SNESIM} simulation only in region \(S^j\).

The \(N_{rot}\) rotation regions and \(N_{aff}\) scaling regions can be independent one from another, thus allowing overlap of rotation regions with scaling regions. Hence the total number of new training images after scaling and rotation is \(N_{rot} \cdot N_{aff}\), and a total of \(N_{rot} \cdot N_{aff}\) different search trees are built and stored prior to simulation. Given one single input training image \(TI\), each new training image \(TI_{i,j}\) associated with rotation region \(i\) and scaling region \(j\) is denoted as:

\[
TI_{i,j}(u) = \Theta_i \cdot \Lambda_j \cdot TI(u^0),
\]

where:
- \(u\) is the node in the new training image,
- \(u^0\) is the node in the original training image,
- \(\Theta_i\) is the rotation matrix for rotation region \(i\):

\[
\Theta_i = \begin{bmatrix}
\cos r^i & \sin r^i & 0 \\
-\sin r^i & \cos r^i & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

(2)

and \(\Lambda_j\) is scaling matrix for affinity region \(j\):

\[
\Lambda_j = \begin{bmatrix}
f^j_x & 0 & 0 \\
0 & f^j_y & 0 \\
0 & 0 & f^j_z
\end{bmatrix}.
\]

(3)
$N_{rot} = 1$ calls for rotating the TI globally and only once. Similarly, $N_{aff} = 1$ corresponds to a single global scaling of the TI.

The rotation and scaling constraints could be derived from well data (Payen-berg et al., 2000), geological outcrops (Cara and Beatrice, 1994) and seismic interpretations (Mondt, 1993; Isaac and Lawton, 2003).

### 3.2 SNESIM simulation with rotation/affinity

A different search tree $T_{tree_{i,j}}$ must be constructed using template $T_{i,j}$ for each new training image TI$_{i,j}$. During the simulation, SNESIM first identifies the rotation region index $i$ and affinity region index $j$ associated with each simulation node $u$; then it uses the corresponding search tree $T_{tree_{i,j}}$ to retrieve the probability distribution conditional to the local data event $dev_{i,j}(u)$; finally a simulated value is drawn from that conditional distribution.

The corresponding SNESIM algorithm is described in Algorithm 1.

### 3.3 Examples

This section presents two examples showing how the SNESIM algorithm has been modified to account for non-stationary information. The concept of local rotation/affinity is demonstrated with and without conditioning to data.

#### 3.3.1 Example 1: 2D unconditional simulation

Figure 3 shows the classic 2D two facies (channel and mud) training image created by Strebelle (2000), which is of size $250 \times 250$. The facies proportions of the mud background (facies 0) and sand channel (facies 1) are 0.72 and 0.28, respectively.

The simulated reservoir is 2D of size $250 \times 250$; it is divided into 3 scaling regions (Figure 4(a)). The affinity factors are $(2, 2, 1)$, $(1, 1, 1)$ and $(0.5, 0.5, 1)$ for regions S1 to S3, respectively. Hence in the SNESIM realizations, the channel thickness in region S1 is expected to be twice that in the training image; the channel thickness in region S2 is same as that in the training image; and the channels in region S3 are half the size of those in training image.

The simulated field is also divided into 3 rotation regions, see Figure 4(b). Hence in region R1, the main channel orientation is expected to be in $45^0$ from the East. Because the original training image has channels elongated in the East-West direction, the training image is rotated by $45^0$ azimuth angle. Similarly, the
Algorithm 1 SNESIM with rotation/affinity concept

1: Define a search template $T_J$
2: Generate new training images for each rotation region $R_i$ per scaling region $S_j$
3: Choose the number $N_G$ of multiple grids
4: Start at the coarsest grid $G^g$, $g = N_G$.
5: while $g > 0$ do
6:   Build a new template $T_j^g$ by rescaling template $T_J$.
7:   for Each rotation region $i$ do
8:     for Each affinity region $j$ do
9:       Use template $T_j^g$ to construct a search tree $Tree_{i,j}$ for training image $TI_{i,j}$
10:      end for
11:    end for
12:    Relocate hard data to the nearest simulation grid nodes
13:    Define a random path visiting all locations to be simulated over all regions
14:    for Each location $u$ along the random path do
15:      Find the conditioning data event $dev_i(u)$ defined by template $T_j^g$
16:      Locate the region index $(i, j)$ of location $u$
17:      Retrieve the conditional cumulative distribution function $P(Z(u) = k|dev_j(u))$ from the corresponding search tree $Tree_{i,j}$
18:      Draw a simulated value $z^{(s)}(u)$ from that conditional distribution and add it to the data set
19:    end for
20:  Go to next multiple grid, $g = g - 1$
21: end while
Figure 3: 2D two facies channel training image (black: sand; gray: mud)

(a) 3 affinity regions
(b) 3 rotation regions

Figure 4: Affinity and rotation regions
original training image is rotated by 90° azimuth angle in region R3. The channel orientation in region R2 remains the same as provided by the original training image.

*SNESIM* is first run with only the scaling regions, using an 2D isotropic search template containing 60 conditioning nodes. Two realizations are given in Figure 5. It is seen that the channel width decreases from region S1 to region S3 as expected. Note that the channel connectivity across the regions is reasonably well preserved, because neighboring nodes across different affinity regions are used for data conditioning.

![Realization 1](image1.png) ![Realization 2](image2.png)

**Figure 5:** Two *SNESIM* realizations with affinity regions only (black: sand channel; gray: mud background; white: region boundaries)

*SNESIM* is then run with 3 rotation regions using the same search template. Figure 6 gives two realizations. It is seen from these *SNESIM* realizations that the channel orientation varies from region R1 to region R3 as expected. Also the channels between regions are reasonably well connected.

Finally, both affinity regions (Figure 4(a)) and rotation regions (Figure 4(b)) are considered jointly for unconditional *SNESIM* simulation. Figure 7 gives two realizations, which all show the width of sand channels varying from one affinity region to another, and the channel orientation varying from one rotation region to another. The channels are reasonably connected across the region boundaries. These two realizations may depict the fan-type deposit shown in Figure 1(a).
Figure 6: Two SNESIM realizations with rotation regions only (black: sand channel; gray: mud background; white: region boundaries)

Figure 7: Two SNESIM realizations with both affinity and rotation regions (black: sand channel; gray: mud background; white: region boundaries)
Note that there is only 7 independent regions due to the overlapping between the rotation regions and affinity regions. However, all $9 = 3 \times 3$ search trees must be built and saved in memory for each multiple grid, because the code does not know in advance the total number of independent regions.

### 3.3.2 Example 2: 2D conditional simulation

In this example, *SNESIM* is performed to simulate the same deltaic fluvial fan reservoir but conditioned to both well hard data (Figure 8(a)) and soft data (Figure 8(b)). There is a total of 50 well facies data. The soft sand probability field was interpreted from seismic data.

![Figure 8: Conditioning data for deltaic fan simulation](image)

The deltaic fan deposit has both affinity and rotation constraints (Figure 4): the affinity regions constrain the channel size and the rotation regions control the channel orientation. *SNESIM* is run with a 60-nodes search template. The tau model was used to integrate the different data sources (Journel, 2002; Remy et al., 2008). Here equal weights ($\tau_1 = \tau_2 = 1.0$) are assigned to the seismic data and the training image. Four conditional realizations are shown in Figure 9. It is seen that the width of channels decreases from the upper-left corner to the lower-right corner; the channel orientation varies from one rotation region to another; and the channels are reasonably connected across the region boundaries. For each realization of Figure 9, it took an average of 57 seconds and 178 MB peak memory on an IBM notebook with 1.3GHz CPU processor.
Figure 9: Four SNESIM realizations with both affinity and rotation conditional to both well hard data and seismic soft data (black: sand channel; gray: mud background; white: region boundaries)
For comparison, one realization generated with the original rotation/affinity concept implemented by Zhang (2002) is given in Figure 10. That realization took 510 seconds (8.5 minutes), and the memory cost is much more than 178 MB. In terms of quality, the four realizations in Figure 16(a) are equally good as the realization in Figure 10. In terms of speed, the new implementation of rotation/affinity concept is significantly CPU-efficient: only about 1/10 of the original CPU time for this 2D example.

![Figure 10: One SNESIM realization using Zhang (2002) implementation of rotation/affinity concept, conditional to both well hard data and seismic soft data (black: sand channel; gray: mud background; white: region boundaries)](image)

Note that although all examples given here are in 2D, the rotation/affinity concept is specifically designed for 3D simulations. It has been applied successfully to generate the second layer of Stanford VI reservoir which is a 3D deltaic fan deposit, of size $150 \times 200 \times 80$ (Castro et al., 2005). The number of rotation regions was 10 and number of affinity regions was 3, for a total of 30 independent regions.

4 **SNESIM Region Concept**

The previous implementation of rotation/affinity concept is CPU-efficient, but can only deal with non-stationary structures that can be made stationary by rotation and affinity. Moreover, it is RAM-demanding because of the large number of search trees that must be kept in memory.
Most often, a subsurface reservoir can be seen as made of several sub-domains, each approximately stationary and associated with its own specific geological features. For example, each of the pseudo regions created from rotation and scaling constraints in Section 3 can be seen as such a sub-domain (Arpat, 2004). One can then model the petrophysical property distribution independently within each sub-domain. Those sub-domains, or regions, can be generated dynamically based on flow response (Caers, 2003; Hoffman and Caers, 2005), or their boundaries can be provided by geologists using seismic data (Cara and Beatrice, 1994; Isaac and Lawton, 2003).

The region concept is proposed to account for the general non-stationary situation when the geological structures differ significantly from one region to another calling for different training images, see Figure 11. Note that the regions can be of any shape, even made of non-connected parts, see region R2. Also, a region can be inactive (for example region R3 is a non-reservoir area), hence there is no need to perform SNESIM simulation in such region. There is usually a gradual transition of facies or properties between regions, calling for data sharing across borders. The region concept allows for such gradual transitions as opposed to a set of totally independent simulations.

Figure 11: Region concept and different training image associated with each region
4.1 Region simulation

The new SNESIM code allows for simulation in each active region with its specific training image and its own parameter settings.

The simulation grid $G$ is first divided into $N_R$ sub-domains (regions) $G^i$, $i = 1, \ldots, N_R$, and $G^1 \cup \cdots \cup G^{N_R} = G$. Then the usual SNESIM simulation is performed in each active region. The regions can be simulated one at a time in any sequence, or they can be simulated simultaneously through a single random path crossing all regions. For example, one can split region R2 on Figure 11 into two parts: R2a (left part) and R2b (right). Then simulate region R2a with a four facies training image followed by a second SNESIM run on region R2b with the same training image. Or, one can simulate regions R2a and R2b simultaneously (one SNESIM run) with the same four facies training image.

After simulation over the first region, simulation in any subsequent region is conditioned to any close-by values previously simulated in neighboring regions. This reduces discontinuities of geological structures across region boundaries. The simulated result not only contains the property values in the current region, but also the property copied from the other conditioning regions. For instance in Figure 11 when the second in sequence region R1 is simulated conditional to the property $r_{e0}$ in region R0, the simulated realization $r_{e0,1}$ contains the property in both R0 and R1. Next the property $r_{e0,1}$ can be used as conditioning data to perform SNESIM simulation in the third in sequence region R2, resulting in property $r_{e0,1,2}$. Finally the property $r_{e0,1,2}$ can be used as conditioning data to perform simulation in the last in sequence region R3, which will result in a realization over all the active areas (R0+R1+R2+R3).

With the region concept the regions are processed one at a time in sequence, consequently one single search tree need to be saved in memory at any time. This significantly relieves the memory demand.

Note that when the training images have different numbers of categories from one region to another, the coding must be exactly the same for the same facies categories in order to ensure simulation consistency among regions. For example, the mud facies in all the training images must have the same code ‘0’, the sand channel the same code ‘1’, etc..

This region concept can be combined with the previous rotation/affinity concept for greater flexibility: each active region can have its own local scaling and orientation constraints. One can also use the region concept to accomplish the tasks of rotation/affinity presented in Section 3:

- first, define all the independent regions from the local rotation and scaling
constraints;

- second, create a new training image for each region;
- last, perform $SNESIM$ simulation sequentially over all the regions.

One can use Python (http://www.python.org/) scripts to automate the simulation tasks.

4.2 2D Examples

$SNESIM$ is run for both unconditional and conditional simulations over a 2D grid. The first example illustrates the general $SNESIM$ region simulation with different training images; the next two examples demonstrate utilization of the region concept to simulate a deltaic fan reservoir

4.2.1 Example 1: 2D unconditional simulation

In this example, $SNESIM$ is performed without any conditioning data over a 2D grid of size $150 \times 100$, which was divided into four regions (Figure 11). Each region is simulated with 60 conditioning nodes in an isotropic search template and 3 multiple grids.

Simulation proceeds as follows:

1. Region R0 is simulated with a three facies channel training image. The training image size is $195 \times 150$, its facies proportions are 0.63, 0.33 and 0.04 for facies category 0 to 2.

2. Region R1 is simulated with a two facies channel training image, with conditioning to previously simulated values in region R0. The training image size is $250 \times 250$, its facies proportions are 0.72 and 0.28 for facies category 0 and 1.

3. Region R2 has two separated parts and is simulated with a four facies training image conditional to simulated values in both R0 and R1. The training image size is $150 \times 150$, its facies proportions are 0.45, 0.20, 0.20 and 0.15 for facies category 0 to 3.

4. Region R3 is simulated with a 5 facies training image conditional to simulated values in regions R0, R1 and R2. The training image size is $123 \times 87$,
its facies proportions are 0.06, 0.34, 0.38, 0.13 and 0.09 for facies category 0 to 4.

Figure 12 shows the simulation evolution as \textit{SNESIM} proceeds from region R0 to region R3. The final simulated realization is also given in Figure 13 with the white lines delineating the region boundaries. It is seen that the continuity of the geological structures across the region boundaries is reasonably well preserved.

The facies coding is consistent among the four training images used, with mudstone coded as category ‘0’, sandstone coded as category ‘1’.

To restate that non-stationary patterns can NOT be captured directly by mp algorithms, the image in Figure 13 is later used as a non-stationary training image to perform \textit{SNESIM} simulation directly. One such resulting realization is given in Figure 14. It is seen that the non-stationary patterns taken from the training image Figure 13 are patched all over the simulation area.

![Simulation in region R0](image1)
(a) Simulation in region R0

![Simulation in region R1](image2)
(b) Simulation in region R1

![Simulation in region R2](image3)
(c) Simulation in region R2

![Simulation in region R3](image4)
(d) Simulation in region R3

Figure 12: \textit{SNESIM} evolutional simulations with 4 regions
Figure 13: Final SNESIM realization with region concept (white lines are the region boundaries)

Figure 14: One SNESIM realization with the non-stationary training image of Figure 12(d)
4.2.2 Example 2: 2D conditional simulation of a fan deposit

In this example, the region concept is used to perform SNESIM simulations accounting for the local rotation and scaling constraints required by a two facies deltaic fan deposit. The original training image is given again in Figure 3. The whole simulation grid (of size $250 \times 250$) is divided into three affinity regions (Figure 4(a)) and three rotation regions (Figure 4(b)). The affinity factors and the rotation azimuth angles are the same as those used for the Example 4 of Section 3.3.

In order to run SNESIM with the region concept, besides defining the previous pseudo regions one must build one new training image for each pseudo region. Due to the overlapping between rotation regions and affinity regions, the total number of independent pseudo regions is only 7. These regions and their associated training images are given in Figure 15, with TI$_4$ being the original input training image. The rotation and scaling constraints for the pseudo regions are given in Table 1.

Figure 15: Seven pseudo regions and the corresponding training images
Table 1: Affinity factors and rotation angles for seven pseudo regions

<table>
<thead>
<tr>
<th>region index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>affinity ( (f_x, f_y) )</td>
<td>( (2, 2) )</td>
<td>( (1, 1) )</td>
<td>( (0.5, 0.5) )</td>
<td>( (1, 1) )</td>
<td>( (0.5, 0.5) )</td>
<td>( (1, 1) )</td>
<td>( (0.5, 0.5) )</td>
</tr>
<tr>
<td>rotation angle</td>
<td>( 45^\circ )</td>
<td>( 45^\circ )</td>
<td>( 45^\circ )</td>
<td>( 0^\circ )</td>
<td>( 0^\circ )</td>
<td>( 90^\circ )</td>
<td>( 90^\circ )</td>
</tr>
</tbody>
</table>

Then *SNESIM* simulation is performed in sequence from pseudo region R1 to region R7, each region with its own training image, and with conditioning data coming from the whole simulation grid. Figure 16 shows four final realizations, all of them are conditioned to the well hard data shown in Figure 8(a) and the seismic soft data shown in Figure 8(b). It is seen that the width of channels decreases from upper fan (upper-left corner) to lower fan (lower-right corner); the channel orientation varies from one rotation region to another; and the channel connectivity is reasonably preserved across the region boundaries. All these realizations are deemed equally good as the realizations of Figure 9 obtained by direct correction for local rotation and affinity.

For each of the realizations of Figure 16 it took only 35 seconds and 24 MB peak memory in the same IBM notebook. Recall that in Example 2 of Section 3.3 with rotation/affinity concept, it took 57 seconds and 178 MB peak memory for each simulation. Hence the concept of region simulation provides gains in terms of both memory reduction and CPU speedup.

The speedup is due in part to more efficient local data conditioning during the region simulations. When searching for the probability distribution in the search tree, for each node \((u + h)\) NOT informed in the local data event \(dev(u)\), the *SNESIM* algorithm looks for all possible data values at that specific search template location \(h\), where \(h\) is the relative offset from the search template center. Hence *SNESIM* needs to traverse and trace down all \(K\) possible tree branches starting from the tree level \(j - 1\), if \(K\) is the total number of categories. This process is very CPU-intensive when many uninformed nodes are found in the data event \(dev(u)\). The *SNESIM* implementation with rotation/affinity concept uses a purely random path over the whole simulation grid, which leads to many uninformed data locations in any conditioning data event \(dev(u)\). With the region simulation concept, the random path is limited to stay within the current active working region. Because all the previously simulated region(s) are fully informed before proceeding to the next region, there are more nodes informed during the simulation in the current working region, resulting in a much lower CPU cost.
Figure 16: Four SNESIM realizations with region concept conditional to both well hard data and seismic soft data (black: sand channel; gray: mud background; white: region boundaries)
4.2.3 Example 3: Combining region concept with rotation/affinity

This example relates to a SNESIM simulation performed only in the active reservoir sub-domains. Figure 17(a) shows a reservoir of size $250 \times 250$ divided into two physical parts: active reservoir part (gray) and inactive non-reservoir part (white). SNESIM simulation is performed only in the active region.

![Active and inactive regions](image)

**Figure 17:** Combination of physical regions with pseudo regions

The region concept can be used together with the local rotation and scaling constraints. In this example, the active region is further constrained by three rotation regions (Figure 4(b)) and three affinity regions (Figure 4(a)), which results in 7 independent pseudo regions (Figure 17(b)). The affinity factors and rotation angles were given in Table 1.

Similar to Example 2, SNESIM simulation is performed in sequence from region R1 to region R7 with conditioning to the well hard data shown in Figure 8(a) and the seismic soft data shown in Figure 8(b). Figure 18 shows two such realizations which reasonably reflect a delta deposit. The channel continuity is reasonably preserved across the pseudo region boundaries although there is no simulated value in the inactive area. Not simulating inside the inactive regions does not affect continuity of the channels.

Next, SNESIM is run again within the inactive regions to fill in the uninformed locations in Figure 18 using the same training images and parameter settings. All the simulated values in Figure 18 are now used for data conditioning. Figure 19...
Figure 18: *SNESIM* region simulations in the active region, conditioning to local rotation and scaling constraints (black: sand channel; gray: mud background; white: inactive regions)

Figure 19: *SNESIM* fill-in simulations with region concept, conditioning to local rotation and scaling constraints (black: sand channel; gray: mud background; white: region boundaries)
shows one fill-in realization for each of two realizations of Figure 18. The original channels of Figure 18 are seen to be reasonably well connected across the new regions being simulated, with correct channel orientation and size. This fill-in simulation took only 14 seconds with a peak 16 MB RAM demand in the IBM notebook. This fast simulation is due to SNESIM being performed only inside the new regions with only one search tree.

4.3 Impact of simulation sequence

Once the simulation grid is divided into \( N_R \) regions, the SNESIM algorithm proceeds from one region to another sequentially. Although simulated values in the regions already processed are used to constrain the simulation in regions processed later, it is important to check the impact of the chosen sequence of region simulations.

The data set in Example 2 of Section 4.2 is considered in this section, which contains both well hard data (Figure 8(a)) and a soft probability field (Figure 8(b)) interpreted from seismic surveys. The training image is a 2D two facies channel training image (Figure 3). The simulation grid is of size \( 250 \times 250 \) divided into 7 regions as shown in Figure 15. The rotation angles and the affinity factors are given in Table 1.

These regions (Figure 15) are simulated with 6 different sequences resulting in 6 different realizations. Table 2 gives the six simulation sequences. For instance, simulation 1 is performed with the region sequence 1, 2, \( \cdots \), and finally 7. The first two simulations started from region (1) with the thickest channels; the next two simulations started from region (2) with medium thickness channels; and the last two simulations started from region (3) with the thinnest channels.

<table>
<thead>
<tr>
<th>simulation #</th>
<th>region sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 2, 3, 4, 5, 6, 7</td>
</tr>
<tr>
<td>2</td>
<td>1, 4, 2, 6, 5, 3, 7</td>
</tr>
<tr>
<td>3</td>
<td>2, 1, 7, 5, 6, 4, 3</td>
</tr>
<tr>
<td>4</td>
<td>2, 5, 7, 3, 4, 6, 1</td>
</tr>
<tr>
<td>5</td>
<td>3, 2, 1, 5, 4, 7, 6</td>
</tr>
<tr>
<td>6</td>
<td>3, 5, 7, 2, 4, 6, 1</td>
</tr>
</tbody>
</table>

Table 2: Region simulation sequence for each simulation
First, \textit{SNESIM} simulation is performed without any conditioning data. The 6 realizations are given in Figure 20. It is seen that:

- realizations 1 and 2 are similar to each other, and channel continuity across regions is reasonably preserved;
- realizations 3 and 4 are also similar, however there appear discontinuities between affinity region 2 and regions 1 and 3;
- realizations 5 and 6 are again similar, and there appears some discontinuities across the affinity region boundaries.
- there are some isolated channels in pseudo regions 3, 5 and 7.

Next, \textit{SNESIM} is run with conditioning only to the well hard data shown in Figure 8(a). The six realizations are given in Figure 21. The channel connectivity between the affinity regions 1 and 2 is improved significantly in realizations 3 and 4. The channel connectivity in regions 3, 5 and 7 is also improved.

Last, \textit{SNESIM} region simulation is run conditioned to both the well hard data shown in Figure 8(a) and the soft probability field shown in Figure 8(b). The resulting six realizations are given in Figure 22. All six realizations are not significantly different, although their simulation sequence is different. The channel continuity is now reasonably well preserved across all region boundaries.

Based on these sensitivity runs, the following conclusions can be made:

- If the region simulation is performed with no conditioning data, the simulation sequence may have significant impact on the final realizations;
- For unconditional simulation with affinity constraints, it is better to start \textit{SNESIM} simulation in those regions with a large affinity factor (thickest patterns);
- Conditioning data, especially soft conditioning data covering the entire study area, reduce the impact of the simulation sequence and improve continuity across regions.

The Python script of \textit{SNESIM} region simulation for this section is given in the Appendix.
Figure 20: Six unconditional SNESIM realizations using region concept with different simulation sequences. For each realization, the number sequence in parentheses indicates the simulation sequence (black: sand channel; gray: mud background)
Figure 21: Six SNESIM realizations using region concept with different simulation sequences, conditioning to well hard data (Figure 8(a)). For each realization, the number sequence in parentheses indicates the simulation sequence (black: sand channel; gray: mud background)
Figure 22: **SNESIM** realizations using region concept with different simulation sequences, conditioning to both well hard data (Figure 8(a)) and soft probability field (Figure 8(b)). For each realization, the number sequence in parentheses indicates the simulation sequence (black: sand channel; gray: mud background).
4.4 3D conditional simulation

In this last example, the *SNESIM* region simulation concept is applied to the 3D Stanford VI data set (Castro et al., 2005), using a large 3D two facies training image (Figure 23). The training image size is $250 \times 250 \times 20$, with the channels elongated in the East-West direction. The training image net-to-gross ratio is 0.30.

![Figure 23: 3D two facies channel training image](image)

The simulation grid is of size $150 \times 200 \times 80$, with 2.4 millions nodes, divided into 3 affinity regions (Figure 24(a)) and 10 rotation regions (Figure 24(b)). The affinity factors are given in Table 3 and the rotation azimuth angles are given in Table 4.

<table>
<thead>
<tr>
<th>region index</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>affinity $(f_x, f_y, f_z)$</td>
<td>(2, 2, 2)</td>
<td>(1, 1, 1)</td>
<td>(0.5, 0.5, 0.5)</td>
</tr>
</tbody>
</table>

Table 3: Affinity factors for Stanford VI reservoir

<table>
<thead>
<tr>
<th>region index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle (degree)</td>
<td>-153</td>
<td>-139</td>
<td>-125</td>
<td>-111</td>
<td>-97</td>
<td>-83</td>
<td>-69</td>
<td>-55</td>
<td>-41</td>
<td>-27</td>
</tr>
</tbody>
</table>

Table 4: Rotation azimuth angles for Stanford VI reservoir
(a) Three affinity regions

(b) Ten rotation regions

Figure 24: Stanford VI affinity and rotation regions

Figure 25: Well hard data
The simulation is performed with 80 conditioning nodes within a 3D search ellipsoid defined by the three major axis ranges 10, 10 and 5. The number of multiple grids is 3. The target sand facies proportion is 0.4. The subgrid option is used with 4 additional conditioning nodes \((J' = 4)\). The simulation is conditioned to 26 wells, with a total of 8,565 samples and a net-to-gross ratio 0.45, see Figure 25. Conditioning also includes a soft probability cube for sand facies as calibrated from seismic data; the average soft probability is 0.42, see Figures 26 and 27. The tau model (Journel, 2002; Remy et al., 2008) is used to integrate that soft probability with unit tau parameter values \(\tau_1 = \tau_2 = 1.0\), giving equal weight to the training image and the soft probability data.

Figure 28 gives a 3D view of one SNESIM realization using the region concept. Figure 29 shows three horizontal sections at \(Z=9, 52\) and 74, and two vertical sections at \(Y=75\) and \(X=27\) of that SNESIM realization; the corresponding sand probability sections are given in Figure 27. One can see that the channel width decreases from South to North, with varying azimuth angles. The simulated sand facies proportion is 0.40.

Overall the channel connectivity is reasonably well preserved, see the long continuous channels in horizontal sections \(Z=9\) and \(Z=74\). Even though the two channel segments inside the white ellipse are not connected on the horizontal section \(Z=52\), they are vertically connected. When compared to Figure 27, the simulated channels of Figure 29 appear to be fairly well constrained by the input soft sand probability cube.

The peak memory cost is about 1 GB, and the total CPU time is 2 hours on a DELL workstation with dual Intel(R) Xeon(TM) 2.8 GHz CPU and 4 GB RAM (this DELL machine is almost 3 times slower than the IBM notebook but has much more memory).

Notice that the training image shown in Figure 23 is very large, and many rotation and affinity regions were considered. The SNESIM implementation with only the rotation/affinity concept would not be able to perform this simulation, because the memory allocation would call for about \(1\,\text{GB} \times 3 \times 10 = 30\,\text{GB}\), much beyond the current computer capability.
Figure 26: Soft conditioning data with overall sand probability 0.42

Figure 27: Soft sand probability field in 2D sections
Figure 28: One SNESIM realization with region concept in 3D view

Figure 29: One SNESIM realization with region concept shown as 2D sections
5 FILTERSIM Region Concept

Similar to SNESIM, the region concept is also implemented into FILTERSIM to account for local non-stationarity information. FILTERSIM simulation can be run over different regions, with each region associated with a specific training image and its own parameter settings. By creating pseudo regions, one can also mimic the process of local rotation and scaling for simple non-stationarity cases using only one single input training image.

5.1 Data type for previously simulated nodes

In FILTERSIM, the data found in the course of simulation can be of the three types (see Figure 30): the original hard data, the inner patch data (previously simulated values frozen as hard data) and the other simulated values. Each data type has a different contribution to the distance calculation and prototype selection (Zhang, 2006; Wu et al., 2006). For the simulation in a current active region, the data event may include data previously simulated in other regions (this allows for a smooth transition across region boundaries). What should be the weights given to these various data types?

![Figure 30: Three data types and their weights: hard data (two square points); patched data (inside the inner pink box); and other data (nodes outside of the pink box)](image)

\[ W_1 \geq W_2 \geq W_3 \]

Figure 30: Three data types and their weights: hard data (two square points); patched data (inside the inner pink box); and other data (nodes outside of the pink box)

Figure 31 gives a 2D continuous training image of size 241 × 241, which displays (1) high values along the channels; and (2) a smooth though short transition zone between these high values and the low values of the mud background.
The simulation grid is of size $250 \times 250 \times 1$, with the 3 local scaling constraints and 3 rotation constraints shown in Figure 4. Due to overlapping, there is a total of 7 independent pseudo regions as shown in Figure 32. The rotation and scaling constraints are given in Table 1. FILTERSIM was performed in sequence from region 1 to region 7. The search template size is $11 \times 11$, the inner patch size is $7 \times 7$.

Figure 33 shows three FILTERSIM unconditional realizations, all using the same parameters. All previously simulated values in regions other than that being simulated are considered of one single data type: either hard data (left plot), or patched data (central plot), or other data (right plot). It appears that channel continuity across the regions decreases as the data type changes from hard data, to patched data then to other data.

Hence for region simulation with FILTERSIM, all previously simulated values in other regions should be hard coded as HARD data type, to ensure better continuity of geological structures across regions.

### 5.2 2D Examples

The first example demonstrates a FILTERSIM region simulation using different training images. In the second example, the impact of the chosen region sequence is analyzed.
Figure 32: Seven pseudo regions for local rotation and affinity constraints ($\times N$ means scaling the original geo-bodies by a factor $N$; $\theta^0$ means rotating the geo-bodies by $\theta$ degree)

Figure 33: The impacts of data type for the values previously simulated in other regions. The white lines are the region boundaries

(a) As hard data  
(b) As patched data  
(c) As other data
5.2.1 2D unconditional simulation

The first example used for SNESIM region simulation (Section 4.2) is run with FILTERSIM. The region settings (Figure 11) and their associated training images are the same except that the training image for region ‘R0’ is rotated with a 90° azimuth angle to make the channel orientations in regions ‘R0’ and ‘R1’ significantly different from each other.

The regions are simulated sequentially from ‘R0’ to ‘R3’, with the same search template of size $11 \times 11$ and inner patch $7 \times 7$. Figure 34 shows one such realization, where the continuity of the geological structures across the region boundaries (white lines) is reasonably well preserved.

When Figure 34 is used as a non-stationary training image to perform unconditional simulation directly, the location-specific non-stationary patterns can not be captured by the simulated realization shown in Figure 35; the training patterns are dropped anywhere in the simulation grid. The long continuous channels of Figure 34 are not reproduced in Figure 35.

5.2.2 Impact of simulation sequence

Similar to Section 4.3, the impact of the region simulation sequence is investigated here. The simulation grid is of size $250 \times 250$, divided into 7 regions (Figure 38).
FILTERSIM is run first without conditioning data using the 2D continuous training image given in Figure 36. The 2D search template size is $11 \times 11$, the inner patch size is $3 \times 3$. The number of multiple grids is 3. The data weights are 0.5, 0.3 and 0.2 for hard data, patched data and other data, respectively. Cross partition is used for pattern classification, using 4 bins for the parent partition. Figure 37 gives the 6 resulting non conditional realizations. The continuity of high values across region boundaries is reasonably well preserved. The simulated realizations appear visually different from one row to another, i.e. when the first region simulated is different. The mean and variance of these 6 realizations are given in Table 5. Any two realizations starting with the same region have similar structures and similar statistics.

Next, FILTERSIM is run conditioned to both well hard data (Figure 8(a)) and soft probability data (Figure 8(b)) using the 2D two facies channel training image of Figure 3. The search template is of size $11 \times 11$, with an inner patch size of $7 \times 7$. FILTERSIM is run with 3 multiple grids, and cross partition with 4 bins for parent classification. Six conditional realizations are given in Figure 38 corresponding to the six different region sequences listed in Table 2. Channels across regions appear reasonably well connected in all six realizations, and the
Figure 36: 2D continuous training image (size 150 × 150)

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training image</td>
<td>0.2815</td>
<td>0.0236</td>
</tr>
<tr>
<td>Realization 1</td>
<td>0.3275</td>
<td>0.0161</td>
</tr>
<tr>
<td>Realization 2</td>
<td>0.3168</td>
<td>0.0175</td>
</tr>
<tr>
<td>Realization 3</td>
<td>0.3246</td>
<td>0.0158</td>
</tr>
<tr>
<td>Realization 4</td>
<td>0.3242</td>
<td>0.0154</td>
</tr>
<tr>
<td>Realization 5</td>
<td>0.3190</td>
<td>0.0176</td>
</tr>
<tr>
<td>Realization 6</td>
<td>0.3115</td>
<td>0.0174</td>
</tr>
</tbody>
</table>

Table 5: Statistics of six unconditional *FILTERSIM* realizations
Figure 37: Six unconditional *FILTERSIM* realizations using region concept with different simulation sequences. For each realization, the simulation sequence is given in parentheses.
Figure 38: Six *FILTERSIM* realizations using the region concept with different simulation sequences, conditioned to both well hard data (Figure 8(a)) and soft probability data (Figure 8(b)). For each realization, the simulated sequence is given in parentheses (black: sand channel; gray: mud background)
sand proportion is 0.24 for each of those realizations. These six realizations are similar to each other, both visually and statistically. In this data-conditioned case, the impact of region simulation sequence does not appear to be significant.

### 5.3 3D conditional simulation

In this example, the Stanford VI data set (Castro et al., 2005) presented in Section 4.4 is used again. FILTERSIM simulation is performed with conditioning to the 26 wells shown in Figure 25 and to the soft probability field shown in Figure 26. The 3D training image is that shown in Figure 23 of size $250 \times 250 \times 20$, with a sand proportion 0.30.

The simulation grid is of size $150 \times 200 \times 80$, divided into 3 affinity regions (Figure 24(a)) and 10 rotation regions (Figure 24(b)). The scaling and rotation settings are given in Table 3 and Table 4. The search template size is $17 \times 17 \times 5$, with an inner patch of size is $7 \times 7 \times 3$. The pattern classification method is cross partition using 4 bins for parent partition and 2 bins for children partition. The number of multiple grids is 3, and the simulation was run with all 9 default filters and using the score-based distance. The tau model (Journel, 2002) is used to integrate the soft data with equal weights $\tau_1 = \tau_2 = 1$.

One FILTERSIM simulation with region concept is given in Figure 39, and five sections of this realization are given in Figure 40. As expected, the channel width decreases from South to North with different orientations constrained by the local rotation. The overall sand facies proportion is 0.28, which is close to the sand proportion of the training image (0.30), even though the net-to-gross ratio derived from sand probability field is 0.40. In FILTERSIM, the input soft probability data is not used directly as the probability, but used only for distance calculation to find the closest prototype class. Hence the contribution of soft probability data is reduced. Consequently the simulated channels of Figure 40 are not well constrained by the soft conditioning data, compare to Figure 27. The channel connectivity is poorer than the SNESIM realization shown in Figure 29.

Next, the proportion transform program TRANSCAT (Remy et al., 2008) is used to post-process the FILTERSIM realization of Figure 39 towards a target sand proportion 0.4. The 3D view and five sections are given in Figures 41 and 42. The net-to-gross ratio of the post-processed realization is now 0.37. The noise in the original realization is removed and the channel connectivity is improved slightly.
Figure 39: One conditional *FILTERSIM* realization using the region concept, $NTG = 0.28$

![Diagrams and images]

Figure 40: *FILTERSIM* realization using the region concept shown as 2D sections
Figure 41: Post-processed realization, $NTG = 0.37$

Figure 42: Post-processed \textit{FILTERSIM} realization shown as 2D sections
In terms of CPU cost, this FILTERSIM realization is 9 times slower than the SNESIM simulation, about 22 hours on the same DELL workstation. However, the peak RAM cost is only 400 MB, much less than the 1 GB needed for the SNESIM realization shown in Figure 28.

6 Conclusions

From the above examples and discussions, the following conclusions can be drawn:

- To run the SNESIM algorithm, the input training image must be reasonably stationary. Non-stationary patterns from the training image can NOT be captured by SNESIM alone.

- The traditional rotation and scaling concept is implemented to account for local non-stationary information where the simulated patterns differ only by orientation and size. This new implementation also significantly improves the simulation speed, with a factor up to 15 in a 2D case (much more for 3D simulations).

- The region concept is proposed to handle the general non-stationary situation where the geological structures are significantly different from one sub-domain to another. This region concept allows to use different training images and different parameters for different reservoir sub-domains, within reasonable CPU and RAM demand.

- The region concept can be combined with the rotation/affinity concept for greater flexibility: each physical region can be further divided into rotation and/or affinity sub-regions.

- The region concept can be applied to mimic the functionality of the rotation/affinity concept through a Python script, while keeping its CPU and RAM advantage.

Acknowledgments

This work was supported by Stanford Center for Reservoir Forecasting (SCRF) and Schlumberger Doll Research (SDR) through the summer internship program. Special thanks go to Dr. Claude Signer and Dr. David McCormick from SDR, for their supports and suggestions.
References


Appendix:
Python Script for SNESIM Region Simulation

The following is an example of how to use Python script to perform SNESIM simulations with the region concept as described in Section 4.3. Before running this script, one must have created/imported one training image for each (pseudo) regions. To ensure there is no line breaking for each SGeMS executable commands, the commands are stored in different strings and concatenated by the string operator “+”.

```
# This script is used to perform snesim simulations with region concept #
# to generate a deltaic fan type deposit. The simulation grid is divided #
# into 7 regions with rotation & scaling constraints, with each region #
# associated with its own training image. #
#
# Author: Jianbing WU #
# Stanford University, 01/15/2007 #

# import sgems modulus
from sgems import *

# The time modulus is used to record the CPU time
# One must have standard Python24 installed for this moduli
import time

# This Python script mimic the rotation/affinity concept using region
# implementation. The training image for each region must be provided
# prior to run this script. Current, the training images are put under
# "TI" grid, and are named as "ti#", #=0~6

# Load project
# Note that the full project path should be provided
execute('LoadProject D:/data/fan/region_sim.prj/')

# record starting time
start=time.clock()

# the 6 simulation sequences
# For each run, only one sequence is allowed
# The sequence is activated by removing the front "#"
```
reg_seq=[0, 1, 2, 3, 4, 5, 6]
#reg_seq=[0, 3, 1, 5, 4, 2, 6]
#reg_seq=[2, 1, 0, 4, 3, 6, 5]
#reg_seq=[2, 4, 6, 1, 3, 5, 0]
#reg_seq=[1, 4, 6, 2, 3, 5, 0]
#reg_seq=[1, 0, 6, 4, 5, 3, 2]

# output name, a suffix seq+"__real#" will be added,  
# seq is the simulation sequence from "reg_seq"  
output_name='cond_sim'

# set simulation grid, which also contains soft data and region coding  
sim_grid='sim_grid'

# set up soft conditioning  
use_soft='1'  
soft_data='prop_mud;prop_sand'  
tau_value='1 1'

# set up region simulation  
use_reg='1'  
reg_coding='region'

# set well hard data  
#well_grid='' % if no well data exist  
well_grid='harddata'  
well_data='face_code'

# set training image  
ti_grid='TI'  
ti_name='ti'

# input random number  
rand_nb=211175

# search ellipsoid definition  
bm_cond='60'  

# nb of conditioning data  
template='10 10 1 0 0 0'  

# 3 ranges + 3 angles

# number of multiple grids  
bm_mg='3'

# servo-system value  
servo='0.5'
# target marginal proportions
prop='0.72 0.28'

# number of realizations
nb_real=1

…………………………………………………………………………………………………………………………
# do NOT modify the follows #
…………………………………………………………………………………………………………………………

# temporary output property, will be remove after each realization
tmp_name='sim'

# generate snesim parameter strings
str_run='RunGeostatAlgorithm snesim_std::/GeostatParamUtils/XML::'
str_run=str_run+'<algorithm name="snesim_std"/>
str_sim_grid=' <GridSelector_Sim value="" + sim_grid +"/>
str_nb_real=' <Nb_Realizations value="1"/>
str_marginal=' <Nb_Facies value="2"/> <Marginal_Cdf value=""+prop+"/>
str_temp= ' <Max_Cond value="" + nb_cond +"/>
str_temp=str_temp+ ' <Search_Ellipsoid value=""+template+"/>
str_hard= ' <Hard_Data grid=""+well_grid+"" property=""+well_data+"/>
str_soft=' <Use_ProbField value="" + use_soft +"/>
str_soft=str_soft+ '<ProbField_properties count="2" value=""+soft_data+"/>
str_soft=str_soft+"/> <TauModelObject value=""+tau_value+"/>
str_vert=' <VerticalPropObject value=""/>
str_vert=str_vert+ ' <VerticalProperties count="0" value=""/>
str_aff_rot=' <Use_Affinity value="0"/> <Use_Rotation value="0"/>
str_reg=' <Use_Region value="" + use_reg +"/>
str_reg=str_reg+ '<Region_Indicator_Prop value=""+reg_coding+"/>
str_cmin=' <Cmin value="1"/>
str_mg=' <Nb_Multigrids_ADVANCED value=""+nb_mg+"/>
str_servo=' <Constraint_Marginal_ADVANCED value=""+servo+"/>
str_resim=' <revisit_nodes_prop value="0"/>
str_debug=' <Debug_Level value="0"/>
str_subgrid=' <Subgrid_choice value="0"/>
str_expansion=' <expand_isotropic value="1"/>
str_expansion=str_expansion+ ' <expand_anisotropic value="0"/>
str_expansion=str_expansion+ ' <aniso_factor value=""/>
str_end=' </parameters>

# set output prefix, use same indent space for each loop
op_name=output_name
for j in range(7):  # for each pseudo region
op_name=op_name+str(reg_seq[j])

# loop over each realization to generate a deltaic fan deposit
# by changing the random seed number
for i in range(nb_real):
    
    # set up random number generator seed
    seed=rand_nb+2*(i-1)
    str_seed=' <Seed value="" + str(seed) + "/">'

    # simulation in the first pseudo region
    active_code=reg_seq[0]  # set current working region
    str_sim=' <Property_Name_Sim value=""+tmp_name+str(0)+"">'/
    str_ti=' <PropertySelector_Training grid="" + ti_grid
    str_ti=str_ti+ " property=""+ti_name+str(active_code)+"">'/
    str_active=' <Active_Region_Code value=""+str(active_code)+"">'/
    str_active=str_active+' <Use_Previous_Simulation value="0">'/

    # run snesim for first region. this first region is conditioned to
    # only the original hard data, soft data within the region
    execute(str_run + str_sim_grid + str_sim + str_nb_real + str_seed
    + str_ti + str_marginal + strtempl + str_hard + str_soft
    + str_vert + str_aff_rot + str_reg + str_active + str_cmin
    + str_mg + str_servo + str_resim + str_debug + str_subgrid
    + str_expansion + str_end)

    # loop from region 1 to region 6. each working region is conditioned
    # additionally to the simulated values in previous regions
    for j in range(1, 7):
        
        active_code=reg_seq[j]  # set current working region
        str_sim=' <Property_Name_Sim value=""+tmp_name+str(j)+"">'/
        str_ti=' <PropertySelector_Training grid="" + ti_grid
        str_ti=str_ti+ " property=""+ti_name+str(active_code)+"">'/
        str_active=' <Active_Region_Code value=""+str(active_code)+"">'/
        str_prev=' <Previous_Simulation_Prop value=""+str(prev)+"">'/
        str_prev=str_prev+tmp_name+str(j-1)+"_real0">'/

        # run snesim for each region with cross-region conditioning
        execute(str_run + str_sim_grid + str_sim + str_nb_real + str_seed
        + str_ti + str_marginal + str_tpl + str_hard + str_soft
        + str_vert + str_aff_rot + str_reg + str_active + str_prev
        + str_cmin + str_mg + str_servo + str_resim + str_debug
        + str_subgrid + str_expansion + str_end )
# output simulated realization(s)
execute('CopyProperty sim_grid::' + tmp_name + '6__real0::sim_grid::' + op_name + '__real' + str(i) + '::0::0')

# delete temporary output files
for j in range(7):
    execute('DeleteObjectProperties sim_grid::' + tmp_name + str(j) + '__real0')

# record ending time
end=time.clock()

# output total simulation time
print "Total simulation time is ", end-start, " seconds"