Geostatistical Earth Modeling Software: User's Manual

Nicolas Remy

May 2004

Contents

Section 1

General Overview

GEMS, the Geostatistical Earth Modeling Software, is an example of software built from scratch using the GsTL. The source code of **GEMS** serves as an example of how to use GsTL facilities.

GEMS was designed with two aims in mind. The first one, geared toward the end-user, is to provide a user-friendly software which offers a large range of geostatistics tools: the most common geostatistics algorithms are implemented, in addition to more recent developments such as multiple-point statistics simulation. The user-friendliness of **GEMS** mainly comes from its non-obtrusive graphical user interface, and the possibility to directly visualize data sets and results in a full 3-D interactive environment. The second objective was to design a software whose functionalities could conveniently be augmented. New features can be added into **GEMS** through a system of plug-ins, i.e. pieces of software which can not be run by themselves but complement a main software. In **GEMS**, plug-ins can be used to add new (geostatistics) tools, add new grid data structures (faulted stratigraphic grids for example) or define new import/export filters.

1.1 First Steps with GEMS

1.1.1 A quick tour to the graphical user interface

The graphical user interface (GUI) of **GEMS** is divided into 3 main parts, see Fig. 1.1:

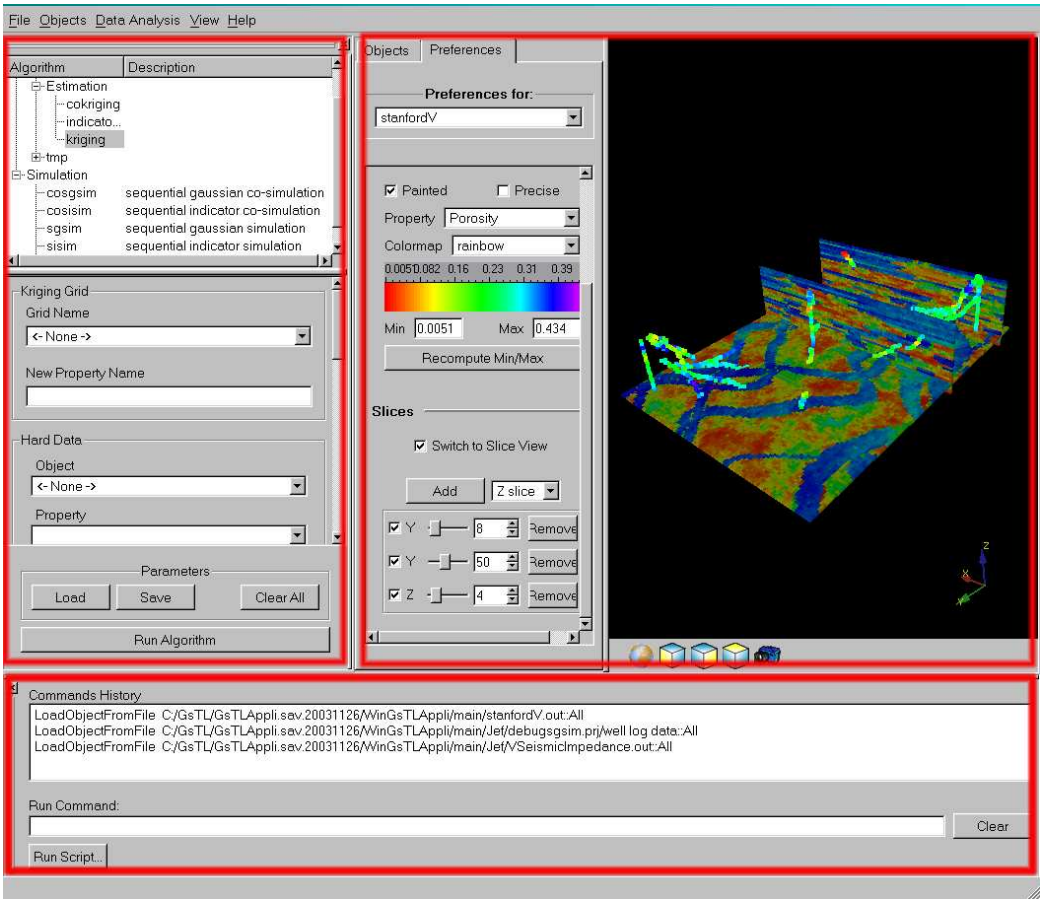


Figure 1.1: **GEMS**'s graphical interface. The three main panels are highlighted in red. The top-right panel is the Algorithm Panel, the top-left is the Visualization Panel and the bottom is the Command Panel

The Algorithm Panel The user selects in this panel which geostatistics tool to use and inputs the required parameters (see Fig. 1.2). The top part of that panel shows a list of available algorithms, e.g. kriging, sequential Gaussian simulation. When an algorithm from that list is selected, a form containing the corresponding input parameters appears below the tools list.

The Visualization Panel One or multiple objects can be displayed in this panel, e.g. a Cartesian grid and a set of points, in an interactive 3-D environment. Visualization options such as color-maps are also set in the Visualization Panel. The Visualization Panel is shown in more details in Fig. 1.3.

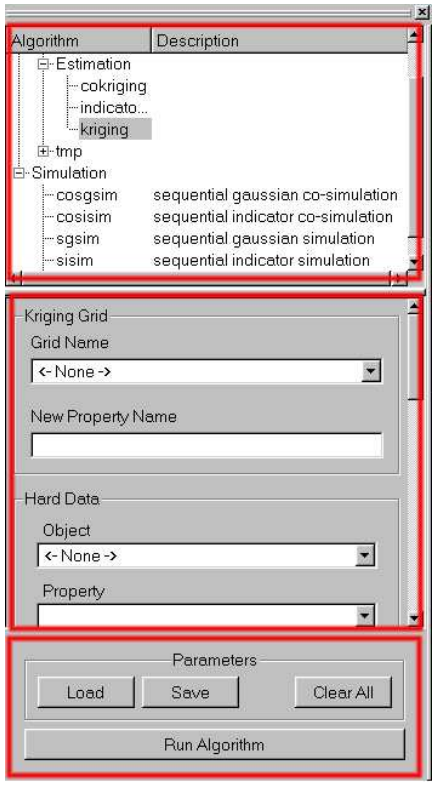


Figure 1.2: The 3 parts of the Algorithm Panel highlighted in red. The top part displays the list of available tools. The middle part is where the input parameters for the selected tool are entered.

The Command Panel This panel is not shown by default when **GEMS** is started. It gives the possibility to control the software from a command line rather that from the GUI. It displays a history of all commands executed so far and provides an input field where new commands can be typed (see Fig. 1.4). See tutorial 1.1.2 for more details about the Command Panel.

1.1.2 A simple tutorial

This short tutorial describes a **GEMS** session in which ordinary kriging is performed on a $100 * 130 * 30$ Cartesian grid. The data consist of a set of 400 points in 3-D space, with a rock porosity value associated to each data point. This point-set object is called `porosity data`.

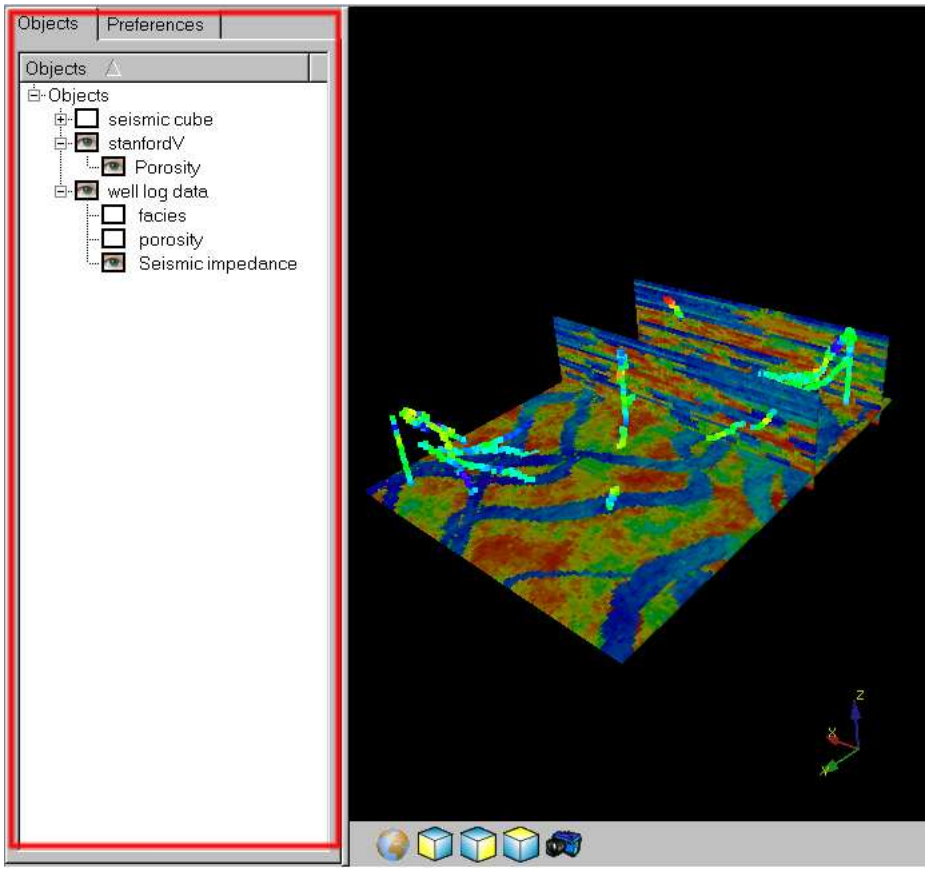


Figure 1.3: The Visualization Panel. The left-hand side (highlighted in red) controls which objects (e.g. grids) are visible. It is also used to set display options, such as which color-map to use.

The steps involved are the following:

1. Load the data set

2. Create a Cartesian grid

3. Select the kriging tool and enter the necessary parameters

4. Display and save the result

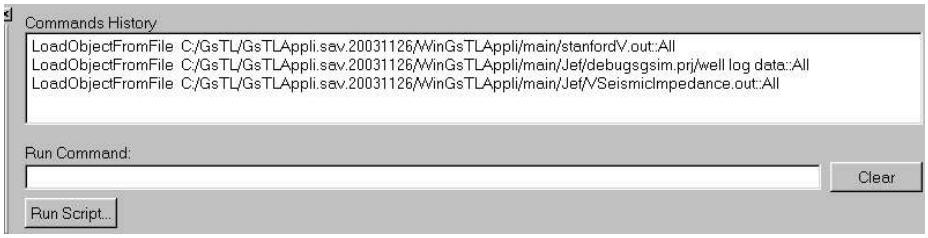


Figure 1.4: The Command Panel

Loading the data set

The first step is to load the data set into the objects database of **GEMS**. Click **Objects** | **Load Object** and select the file containing the data. Refer to section 1.4.1 for a description of the available data file formats. When the object is loaded a new entry called `porosity data` appears in the **Objects** section of the Visualization Panel, as shown in Fig. 1.5.

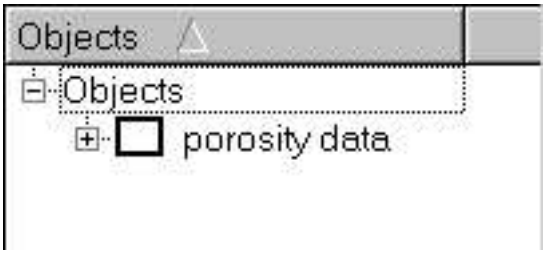


Figure 1.5: Object list after the data set is loaded

Click in the square before the point-set name to display it. Displayed objects have a little eye painted inside the rectangle before their name. The plus sign before the square indicates that the object contains properties. Click on the plus sign to display the list of properties. Click in the square before the property name to paint the object with the corresponding property (see Fig. 1.6).

Creating a grid

The next step is to create the grid on which kriging will be performed. The grid we will create is a 3-D Cartesian grid with $100 * 130 * 30$ cells.

Click **Objects** | **New Cartesian Grid** to open the grid creation dialog. Enter the dimensions of the grid, the coordinates of the origin of the grid, i.e. the lower left corner

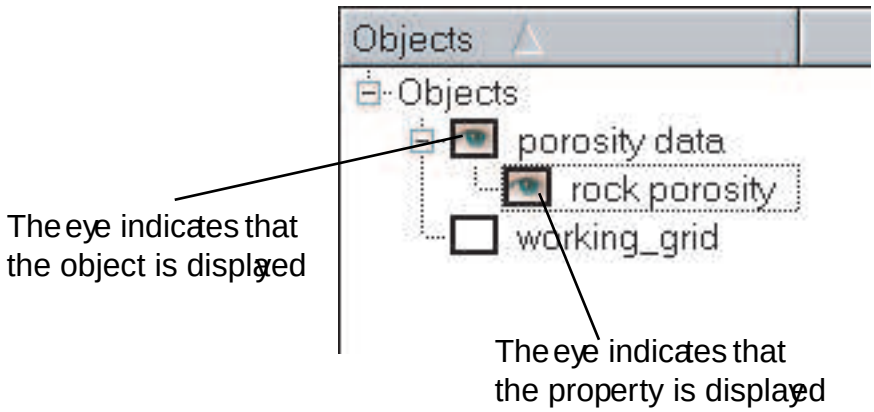


Figure 1.6: Showing/hiding an object or a property

(i.e. the point of the grid with smallest x, y, z coordinates), and the dimensions of each grid cell. Provide a name for the new grid, `working_grid` for example. Click **Create Grid** to create the grid. A new entry called `working_grid` appears in the **Objects** panel of the Visualization Panel, as shown in Fig. 1.7.

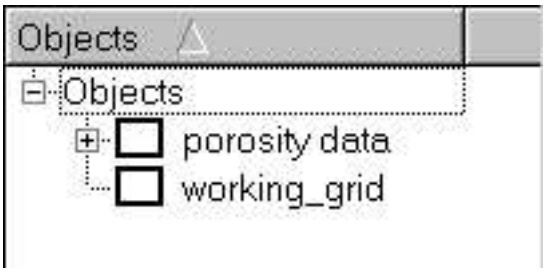


Figure 1.7: Object list after the Cartesian grid is created

The object data-base now contains two objects: a point-set with the rock porosity property, and a Cartesian grid with not yet any property attached. We can proceed to the kriging run.

Running the kriging algorithm

Select the kriging tool from the list in the Algorithm Panel. A form prompting for the kriging parameters appears below the algorithms list. You can either type in the parameters or load them from a file. Fig. 1.8 shows an example of parameter file for kriging (refer

to section 1.4.2 for a description of the parameter file format and to section 2.2 for details on kriging parameters). Using the parameters of Fig. 1.8, ordinary kriging is performed with an isotropic search ellipsoid of radius 50 (section 1.2 describes how to specify a 3-D ellipsoid in **GEMS**) and an isotropic spherical variogram of range 30, sill 1, and a nugget effect of 0.1 (section 1.3 explains how to specify a variogram).

```
<parameters>  <algorithm name="kriging" />
  <Grid_Name  value="working_grid"  />
  <Property_Name  value="krig_porosity" />
  <Hard_Data  grid="porosity data"   property="porosity"  />
  <Kriging_Type  type="Ordinary Kriging (OK)" >
  <parameters />
</Kriging_Type>
  <Max_Conditioning_Data  value="12" />
  <Search_Ellipsoid  value="50 50 50
                            0  0  0 " />
  <Variogram  nugget="0.1" structures_count="1"  >
  <structure_1  contribution="0.9"  type="Spherical"   >
    <ranges max="30"  medium="30"  min="30"   />
    <angles x="0"  y="0"  z="0"   />
  </structure_1>
</Variogram>
</parameters>
```

Figure 1.8: *Kriging* parameter file

Once all parameters have been entered, click the **Run Algorithm** button. If some parameters are not correctly set, they are highlighted in red and a description of the error will appear if the mouse is left a few seconds on the offending parameter.

If kriging was run with the parameters shown on Fig. 1.8, the grid named `working_grid` now contains a new property called `krig_porosity`.

Displaying and saving the result

The algorithm *Kriging* created a new property `krig_porosity` in the grid `working_grid`. Click on the plus sign before the `working_grid` entry in the objects list to show the list

of properties attached to the grid, and click in the square before the newly created property to display it. To save the results, click **Object** | **Save Object** to open the Save Object dialog. Provide a file name, the name of the object to save, e.g. `working_grid` and the file format to use (see section 1.4.1).

It is also possible to save all the objects at once by saving the project (**File** | **Save Project**).

1.1.3 Automating tasks in GEMS

Tutorial 1.1.2 showed how to perform a single run of the kriging algorithm. Next, one would like to study the sensitivity of the algorithm to parameter `Max_Conditioning_Data`, the maximum number of conditioning data retained for each kriging. The user would like to vary that number from 1 to 50 in increments of 1. It would be very impractical to perform such a study in successive sequences as explained in Tutorial 1.1.2.

GEMS provides a solution to this problem through its command line interface. Many actions in **GEMS** can either be performed with mouse clicks or by typing a command in the Command Panel. For example, loading the data set in step 1 of Tutorial 1.1.2 could have been achieved by typing the following command:

```
LoadObjectFromFile  /home/user/data_file.dat:All
```

Each command has the following format:

- the name of the command, e.g. `LoadObjectFromFile`
- a list of parameters, separated by a colon ":". In the previous example two parameters were supplied: the name of the file to load, and the file format `All` (meaning that every available file format should be considered).

Every command performed in **GEMS**, either typed or resulting from mouse clicks, is recorded to both the "Commands History" section of the Command Panel and to a file called `gstlappli_history.log`. Hence if one does not know a command name, one can use the GUI to perform the corresponding action and check the command name and syntax in the command history.

It is possible to combine several commands into a single file and have **GEMS** execute them all sequentially. For the sensitivity study example, one would have to write a "script"

file containing 50 kriging commands, each time changing the `Max_Conditioning_Data` parameter. Note that there are no control structures, e.g. "for" loops, in **GEMS** script files. However there are many programming tools available (Perl, Awk, Shell,...) that can be used to generate such script file.

1.2 Defining a 3-D ellipsoid

Most of the algorithms in **GEMS** require the user to specify a 3-D ellipsoid, to represent a search volume for example, or three anisotropy directions and affinity coefficients. In **GEMS** a 3-D ellipsoid is represented by six parameters: the three dimensions of the ellipsoid major axes $r_{max}, r_{med}, r_{min}$, and three angles, α, β, θ positioning the ellipsoid in space (see Figs. 1.10, 1.11 and 1.12).

Let (X, Y, Z) be the three orthogonal axes of a Cartesian coordinate system. The position of the ellipsoid is obtained by three successive rotations. All angles are measured counter-clockwise. Initially, before any rotation, the major axis of the ellipsoid is aligned with X, the medium axis with Y, and the minor axis with Z, see Fig. 1.9.

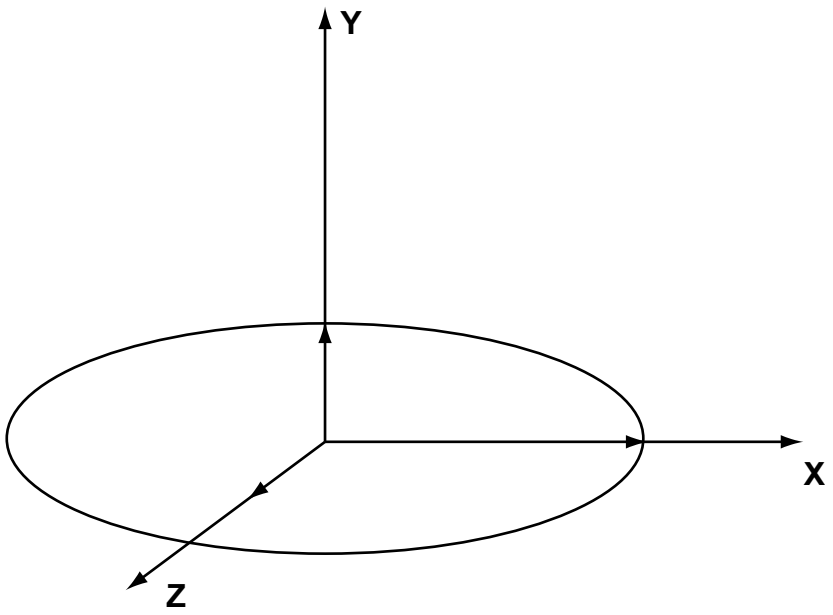


Figure 1.9: Starting point

first rotation about Z: The ellipsoid is first rotated about axis Z, by angle α, typically called *azimuth*. Let (X', Y', Z') be the rotated coordinate system, with $Z' = Z$, see Fig. 1.10.

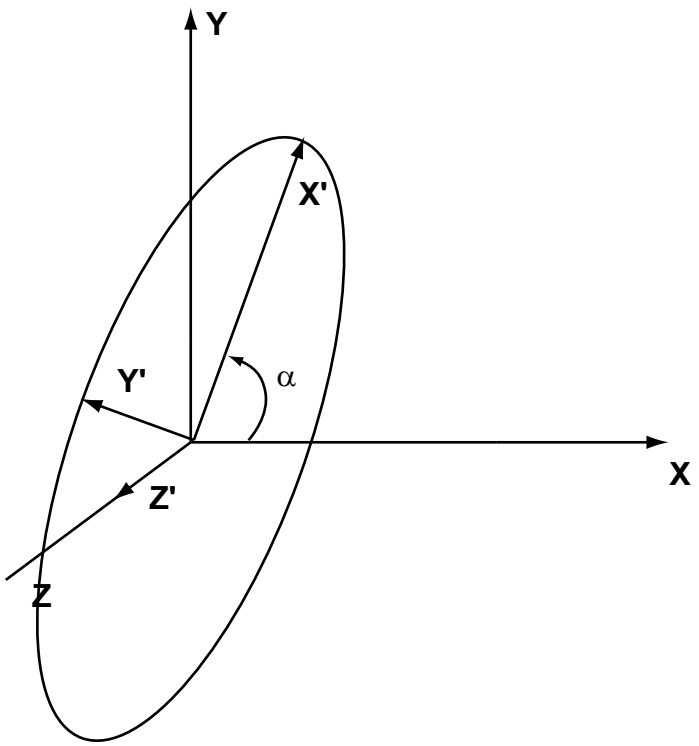


Figure 1.10: First rotation about Z

second rotation about Y': The ellipsoid is then rotated about axis Y', by angle β, called the *dip*. Let (X'', Y'', Z'') be the rotated coordinate system, with $Y'' = Y'$ see Fig. 1.11.

third rotation about X'': Last, the ellipsoid is rotated about axis X'' by angle θ, called the *plunge* or *rake* angle, leading to the final coordinate system (X''', Y''', Z'''), with $X''' = X''$, see Fig. 1.12.

The final transformation matrix, product of three rotations, is given by:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

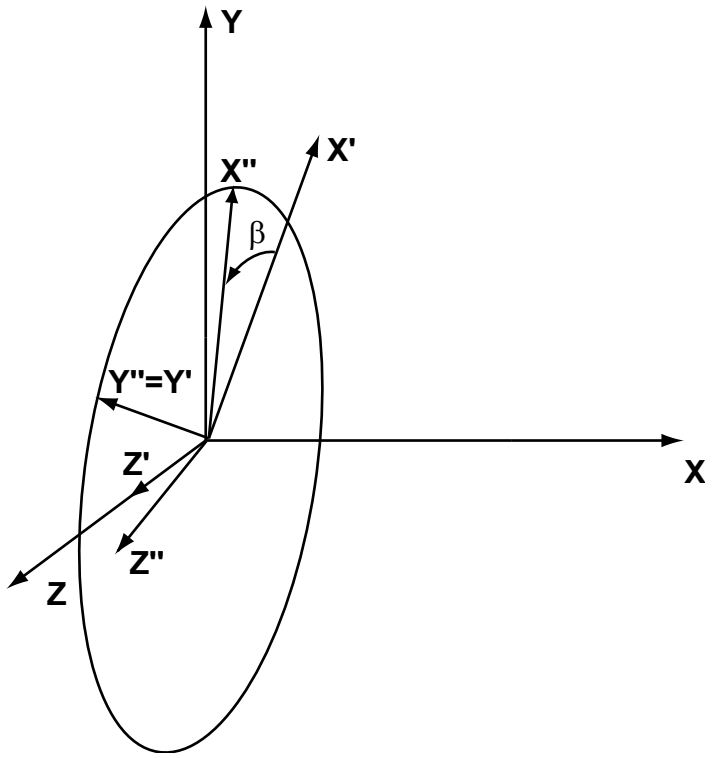


Figure 1.11: Second rotation about Y'

Note that the order in which the rotations are performed is important. In the **GEMS** parameter files, the six parameters defining an ellipsoid must be given in the following order: $r_{max}, r_{med}, r_{min}, \alpha, \beta, \theta$, where r_{max} is the dimension of the major axis, r_{med} the dimension of the medium axis, and r_{min} is the dimension of the minor axis.

Relation to *GSLIB*'s conventions

GSLIB [Deutsch and Journel, 1992] uses a different set of three angles to position the ellipsoid in space. They are related to **GEMS** angles as follows:

$$\begin{aligned} \xi_1 &= 90^o - \alpha \\ \xi_2 &= -\beta \\ \xi_3 &= \theta \end{aligned}$$

where ξ_1, ξ_2, ξ_3 are the angles specified in the *GSLIB* parameter files, in that order.

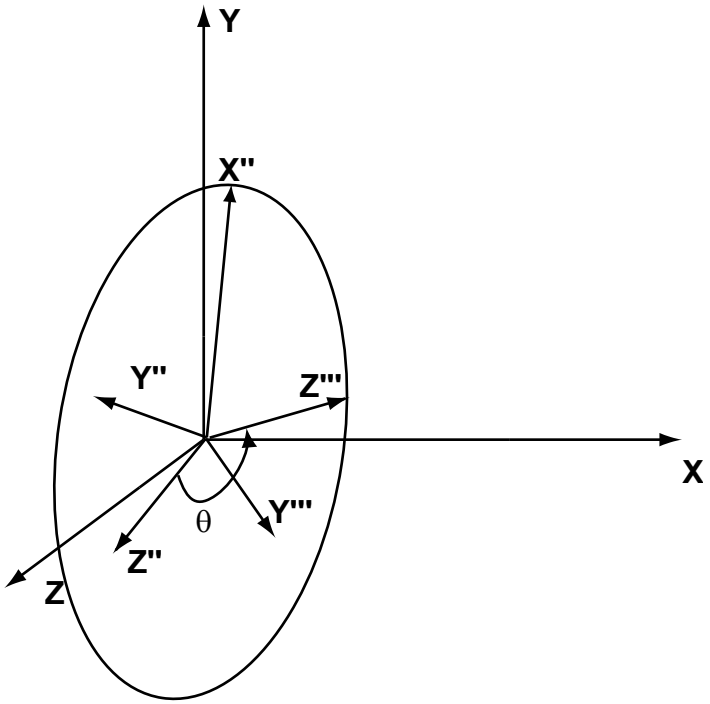


Figure 1.12: Third rotation about X''

1.3 Variogram Model Specification

Variogram models are called by most of **GEMS** algorithms. The current version of **GEMS** allows four basic models and any positive linear combinations of them, that is the linear model of regionalization, [Goovaerts, 1997a]. The four basic isotropic models are: the nugget effect model, the spherical model, the exponential model and the Gaussian model.

nugget effect model

$$\gamma(\mathbf{h}) = \begin{cases} 0 & \text{if } \|\mathbf{h}\| = 0 \\ 1 & \text{otherwise} \end{cases} \tag{1.1}$$

A pure nugget effect model for a variable $Z(\mathbf{u})$ expresses a lack of (linear) dependence between variables $Z(\mathbf{u})$ and $Z(\mathbf{u}+\mathbf{h})$

spherical model with actual range a

$$\gamma(\mathbf{h}) = \begin{cases} \frac{3}{2}\frac{\|\mathbf{h}\|}{a} - \frac{1}{2}(\frac{\|\mathbf{h}\|}{a})^3 & \text{if } \|\mathbf{h}\| \leq \mathrm{a} \\ 1 & \text{otherwise} \end{cases} \tag{1.2}$$

exponential model with practical range a

$$\gamma(\mathbf{h}) = 1 - \exp\left(\frac{-3\|\mathbf{h}\|}{\mathrm{a}}\right) \tag{1.3}$$

Gaussian model with practical range a

$$\gamma(\mathbf{h}) = 1 - \exp\left(\frac{-3\|\mathbf{h}\|^2}{\mathrm{a}^2}\right) \tag{1.4}$$

All these models are permissible in 3-D and have a covariance counterpart:

$$\gamma(\mathbf{h}) = \mathrm{C}(\mathbf{0}) - \mathrm{C}(\mathbf{h})$$

For this reason, we will interchangeably refer to either the covariance or the variogram in the remainder of the text.

In **GEMS** a variogram model: $\gamma(\mathbf{h}) = \mathrm{c}_0\gamma^{(0)}(\mathbf{h}) + \sum_{\mathrm{l}=1}^{\mathrm{L}} \mathrm{c}_\mathrm{l}\gamma^{(\mathrm{l})}(\mathbf{h})$ is characterized by the following parameters:

- a nugget effect $c_0\gamma^{(0)}$ with nugget constant $c_0 \geq 0$
- the number L of nested structures. Each structure $\gamma^{(l)}(\mathbf{h})$ is then defined by:
  - a variance contribution $c_l \geq 0$
  - the type of the variogram: spherical, exponential or Gaussian
  - six parameters: the three practical ranges and the three corresponding rotation angles defining the anisotropy ellipsoid, see Section 1.2. Note that each nested structure can have a different anisotropy.

Example:
Consider a variogram $\gamma(\mathbf{h}) = 0.3 + 0.4\gamma^{(1)}(\mathbf{h}) + 0.3\gamma^{(2)}(\mathbf{h})$, with:

- a nugget constant 0.3

- $\gamma^{(1)}(\mathbf{h})$ an anisotropic spherical variogram with major range 40, medium range 20 and minor range 5, and angles $\alpha = 45^o$, $\beta = 0$, $\theta = 0$

- $\gamma^{(2)}(\mathbf{h})$ an isotropic Gaussian variogram of range 200

Fig. 1.13 shows how this variogram model is input using the graphical interface.

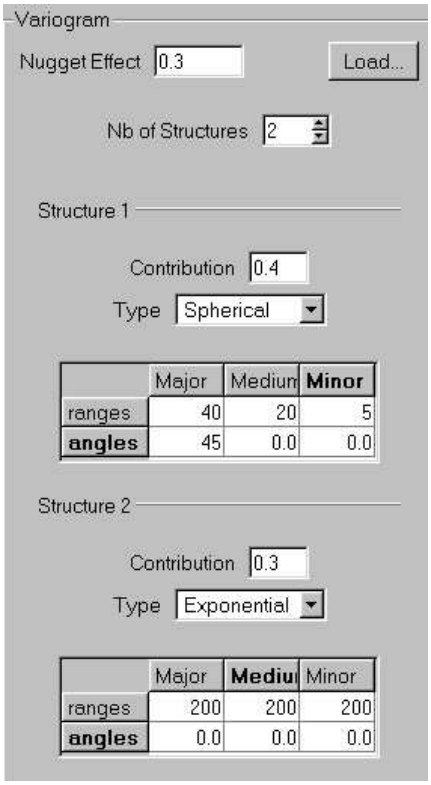


Figure 1.13: Inputing γ using the GUI

The corresponding parameter file snippet would be:

```
<[Parameter_name]  nugget="0.3"  structures_count="2" >
    <structure_1  contribution="0.4"  type="Spherical"   >
        <ranges max="40"  medium="20"  min="5"   />
        <angles x="45"  y="0"  z="0"   />
    </structure_1>
    <structure_2  contribution="0.3"  type="Exponential"   >
        <ranges max="200"  medium="200"  min="200"   />
        <angles x="0"  y="0"  z="0"   />
    </structure_2>
</[Parameter_name]>
```

Where `[Parameter_name]` is the name of the parameter. See Section 1.4.2 for more details on the parameter file format.

1.4 File Formats

1.4.1 Objects Files

GEMS supports two file formats by default to describe grids and sets of points: the *GSLIB* format and the **GEMS** binary format.

The *GSLIB* file format

It is a simple ASCII format used by the *GSLIB* software [Deutsch and Journel, 1992]. It is organized by lines:

- the first line gives the title.
- the second line is a single number n indicating the number of properties in the object.
- the n following lines contain the names of each property (one property name per line)
- each remaining line contains the values of each property (n values per line) separated by spaces or tabulations. The order of the property values along each line is given by the order in which the property names were entered.

Note that the same format is used for describing both point sets and Cartesian grids. When a *GSLIB* file is loaded into **GEMS** the user has to supply all the information that are not provided in the file itself, e.g. the name of the object, the number of cells in each direction if it is a Cartesian grid.

The GEMS binary file format

GEMS can use an uncompressed binary file format to store objects. Binary formats have two main advantages over ASCII files: they occupy less disk space and they can be loaded and saved faster. The drawback is a lack of portability between platforms. **GEMS** binary format is self-contained, hence the user need nnot provide any additional information when loading such a file.

1.4.2 Parameter Files

When an algorithm is selected from the Algorithm Panel (see step 3 of Tutorial 1.1.2), several parameters are called for. It is possible to save those parameters to a file, and later retrieve them.

The format of a parameter file in **GEMS** is based on the eXtended Markup Language (XML), a standard formating language of the World Wild Web Consortium (*www.w3.org*). Fig. 1.8 shows an example of such parameter file.

In a parameter file, each parameter is represented by an XML element. An element consists of an opening and a closing tag, e.g. `<tag>` and `</tag>`, and one or several attributes. Following is an example of an element called `algorithm` which contains a single attribute "`name`":

```
<algorithm  name="kriging"> </algorithm>
```

Elements can themselves contain other elements:

```
<Variogram  nugget="0.1" structures_count="1"  >
    <structure_1  contribution="0.9"  type="Spherical"   >
        <ranges max="30"  medium="30"  min="30">   </ranges>
        <angles x="0"  y="0"  z="0">   </angles>
```

```
    </structure_1>
</Variogram>
```

Here the element `Variogram` contains an element `structure_1`, which itself contains two elements: `ranges` and `angles`. Each of these elements have attributes. Note that if an element only contains attributes the closing tag can be abbreviated: in the previous example, element `ranges` only contains attributes and could have been written:

```
<ranges max="30"  medium="30"  min="30" />
```

The `/>` sequence indicates the end of the element

A **GEMS** parameter file always has the two following elements:

- element `parameters`. It is the root element: it contains all other elements
- element `algorithm`. It has a single attribute `name` which gives the name of the algorithm for which parameters are specified.

All other elements are algorithm-dependent and are described in sections 2 and 3.

Such XML formated parameter file has several advantages:

- Elements can be entered in any order
- comments can be inserted anywhere in the file. A comment block starts with `<!--` and end with `-->`. They can span multiple lines. For example:

```
<!-- An example of a comment block spanning
     multiple lines -->

<parameters>  <algorithm name="kriging" />

  <!-- the name of the working grid -->
  <Grid_Name  value="working_grid"  />
</parameters>
```

1.5 Data Set

Stanford V [Mao and Journel, 1999] is a synthetic fluvial channel clastic reservoir data set. Data on lithofacies, rock porosity, density and sonic velocity are available in the whole reservoir. Thirteen vertical and horizontal wells, drilled from 5 plateforms, are shown on Fig. 1.14.

The *Stanford V* reservoir is divided into three layers, each divided into $100x130x10$ grid cells. In each layer, the channels have a different thickness and orientation. Fig. 1.15 shows some horizontal cross-sections of each layer. Fig. 1.16 shows the corresponding porosity values, as well as the porosity histogram of each layer.

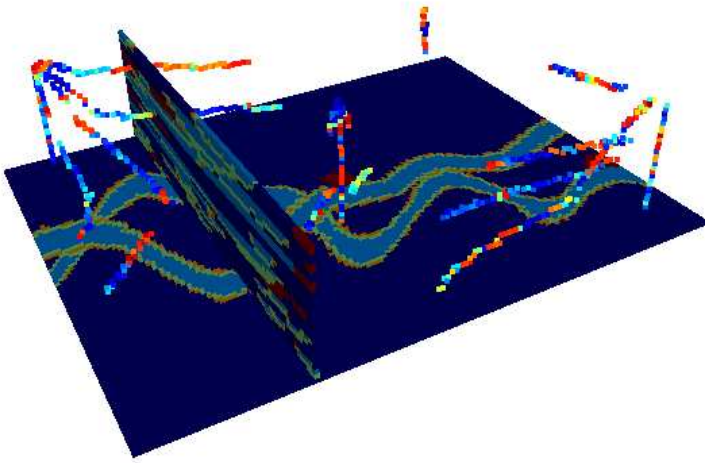
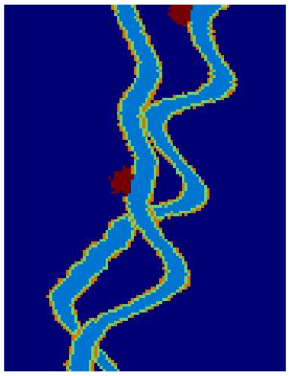
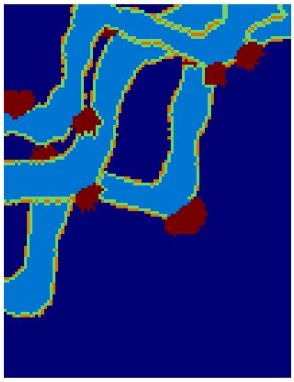


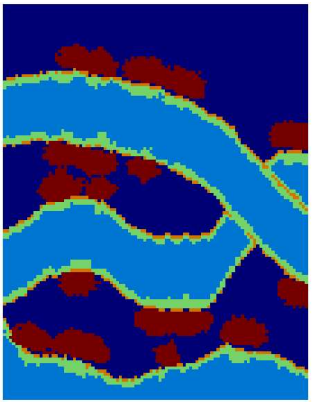
Figure 1.14: Wells and a horizontal cross-section of the *Stanford V* reservoir



(a) Layer 1

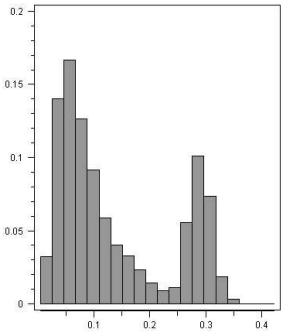


(b) Layer 2

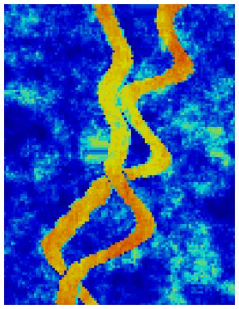


(c) Layer 3

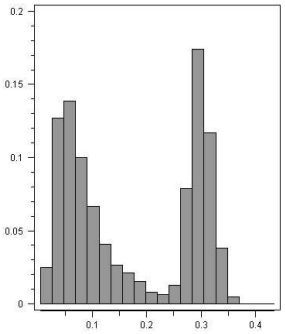
Figure 1.15: Cross-sections of each of the 3 layers showing the lithofacies



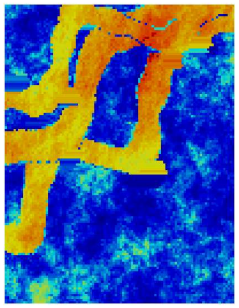
(a) Layer 1



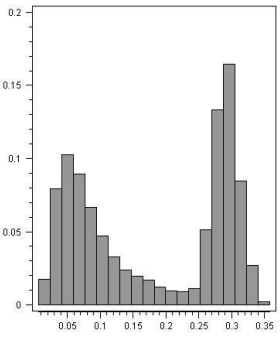
(b) Layer 1



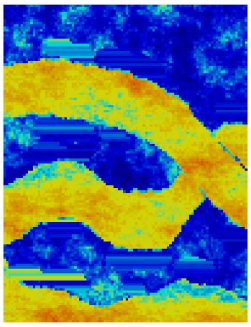
(c) Layer 2



(d) Layer 2



(e) Layer 3



(f) Layer 3

Figure 1.16: Porosity cross-sections and histograms of each of the 3 layers

Section 2

Estimation Algorithms

GEMS provides several tools for estimating a spatially distributed variable $Z(\mathbf{u})$ from a limited number of samples. All algorithms rely on the linear, least-square estimation paradigm called kriging. The kriging estimator can be extended to deal with a variable whose mean is not stationary, and to the case of multiple covariates (*cokriging*).

2.1 Common Parameters

All estimation algorithms in **GEMS** work on a 3-D object, loosely called the estimation grid: in the current version, that object can either be a Cartesian grid or an unstructured set of points. When an estimation algorithm is run on an object, a new property containing the result of the estimation is attached to that object. All estimation algorithms require the following two parameters:

Parameters description

`Grid_Name:` The grid (or more generally, the object) on which the estimation is to be performed

`Property_Name:` The name of the new property resulting from the estimation

2.2 Kriging

Kriging is a 3-D estimation program for variables defined on a constant volume support. Estimation can either be performed by simple kriging (SK), ordinary kriging (OK), kriging with a polynomial trend (KT) or simple kriging with a localy varying mean (LVM). LVM is a variant of kriging, in which it is assumed that the mean $m(\mathbf{u})$ is not constant but is known for every $\mathbf{u}$. The kriging problem then becomes to find the weights $\{\lambda_\alpha\}$, $\alpha = 1, \ldots, n$ such that:

$$Var\Big(\sum_{\alpha=1}^{n} \lambda_\alpha [Z(\mathbf{u}_\alpha) - \mathrm{m}(\mathbf{u}_\alpha)] - [\mathrm{Z}(\mathbf{u}) - \mathrm{m}(\mathbf{u})]\Big) \quad \text{is minimum}$$

Example:
Simple kriging is used to estimate porosity in the second layer of Stanford V. Porosity is modeled by a stationary random function with an anisotropic spherical variogram. Its expected value, inferred from the well data, is 0.17. The input parameters for algorithm *Kriging* are reproduced in Fig. 2.1. Fig. 2.2 shows the estimated porosity field.

Parameters description

`Hard_Data:` The grid and the name of the property containing the hard data

`Kriging_Type:` Possible types of kriging include: Simple Kriging (SK), Ordinary Kriging (OK), Kriging with Trend (KT) and Simple Kriging with Localy Varying Mean (LVM)

`Trend:` The trend components. Possible components are, with $\mathbf{u} = (\mathrm{x}, \mathrm{y}, \mathrm{z})$: $t_1(\mathbf{u}) = \mathrm{x}$, $t_2(\mathbf{u}) = \mathrm{y}$, $t_3(\mathbf{u}) = \mathrm{z}$, $t_4(\mathbf{u}) = \mathrm{x.y}$, $t_5(\mathbf{u}) = \mathrm{x.z}$, $t_6(\mathbf{u}) = \mathrm{y.z}$, $t_7(\mathbf{u}) = \mathrm{x}^2$, $t_8(\mathbf{u}) = \mathrm{y}^2$, $t_9(\mathbf{u}) = \mathrm{z}^2$.

The trend is coded by a string of 9 flags, for example the string "`0 1 0 0 0 1 0 1 0''` would correspond to a trend with components t_2, t_6 and t_8, i.e. $T(\mathbf{u}) = \alpha\,\mathrm{x} + \beta\,\mathrm{yz} + \delta\,\mathrm{y}^2$. Each flag is separated by a space.

```
<parameters>  <algorithm name="kriging" />
  <Grid_Name  value="layer 2"  />
  <Property_Name  value="krig" />

  <Hard_Data  grid="layer 2 well data"   property="porosity"  />

  <Kriging_Type  type="Simple Kriging (SK)" >
    <parameters mean="0.17" />
  </Kriging_Type>

  <Max_Conditioning_Data  value="200" />
  <Search_Ellipsoid  value="100 100 5
                            0 0 0 " />

  <Variogram  nugget="0.2" structures_count="1"  >
    <structure_1  contribution="0.8"  type="Spherical"   >
      <ranges max="80"  medium="40"  min="2"   />
      <angles x="30"  y="0"  z="0"   />
    </structure_1>
  </Variogram>
</parameters>
```

Figure 2.1: Simple kriging parameters

`Local_Mean_Property:` The property of the simulation grid containing the non-stationary mean. A mean value must be available at each location to be estimated.

`Max_Conditioning_Data:` The maximum number of conditioning data to be used for kriging at any location

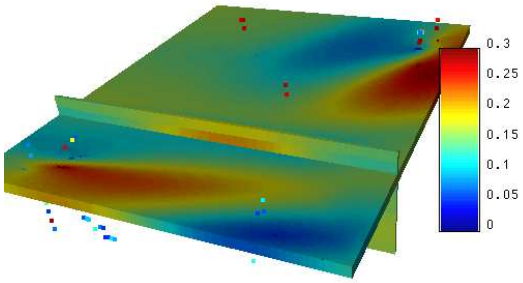
`Search_Ellipsoid:` The ranges and angles defining the search ellipsoid

`Variogram:` The variogram of the variable to be estimated by kriging

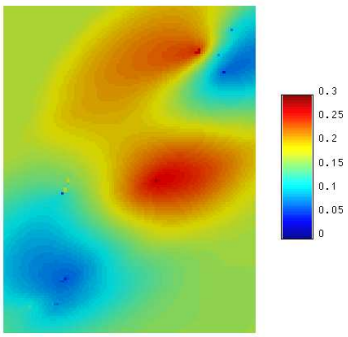
2.3 Indicator Kriging

Let $Z(\mathbf{u})$ be a continuous random variable, and $I(\mathbf{u}, \mathrm{z_k})$ the binary indicator function defined at cutoff z_k:

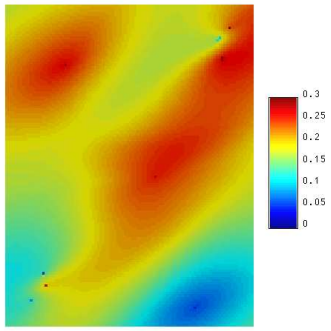
$$I(\mathbf{u}, \mathrm{z_k}) = \begin{cases} 1 & \text{if } Z(\mathbf{u}) \leq \mathrm{z_k} \\ 0 & \text{otherwise} \end{cases}$$



(a) 3D view



(b) Cross-section 1



(c) Cross-section 2

Figure 2.2: Porosity estimated by simple kriging

The aim of indicator kriging is to estimate the conditional cumulative distribution function (ccdf) at any cutoff z_k, conditional to data (n):

$$\begin{aligned} I^*(\mathbf{u}, \mathrm{z_k}) &= E^*\big(I(\mathbf{u}, \mathrm{z_k}) \mid (\mathrm{n})\big) \\ &= Prob^*\big(Z(\mathbf{u}) < \mathrm{z_k} \mid (\mathrm{n})\big) \end{aligned} \tag{2.1}$$

$I^*(\mathbf{u}, \mathrm{z_k})$ is estimated by the simple kriging estimator:

$$I^*(\mathbf{u}, \mathrm{z_k}) - \mathrm{E}\{\mathrm{I}(\mathbf{u}, \mathrm{z_k})\} = \sum_{\alpha=1}^{\mathrm{n}} \lambda_\alpha \Big(\mathrm{I}(\mathbf{u}_\alpha, \mathrm{z_k}) - \mathrm{E}\{\mathrm{I}(\mathbf{u}_\alpha, \mathrm{z_k})\} \Big)$$

Estimating $I^*(\mathbf{u}, \mathrm{z_k})$ for different cutoffs z_k, k =1,...,K, yields a discrete estimate of the conditional cumulative distribution function at the threshold values $z_1, \ldots, z_K$.

The algorithm *Indicator Kriging* assumes that the marginal probabilities $E\{I(\mathbf{u}_\alpha, \mathrm{z_k})\}$ are constant and known for all $\mathbf{u}_\alpha$ and z_k (Simple Indicator Kriging [Goovaerts, 1997a]).

Median IK

Estimating a conditional cumulative distribution function at a given location $\mathbf{u}$ requires the knowledge of the variogram of each of the indicator variables $I(., z_k)$, k =1, ..., K. Inference of these K variograms can be a daunting task. Moreover a kriging system has to be solved for each indicator variable. The inference and computational burden can be alleviated if two conditions are met:

- the K indicator variables are intrinsically correlated [Goovaerts, 1997a]:

$$\gamma_Z(\mathbf{h}) = \gamma_{\mathrm{I}}(\mathbf{h}, \mathrm{z_k}) = \gamma_{\mathrm{I}}(\mathbf{h}, \mathrm{z_k}, \mathrm{z_{k'}}) \quad \forall \mathrm{z_k}, \mathrm{z_{k'}}$$

  where $\gamma_Z(\mathbf{h})$ is the variogram of variable $Z(\mathbf{u})$ and $\gamma_I(\mathbf{h}, \mathrm{z_k}, \mathrm{z_{k'}})$ is the cross-variogram between indicator variables $I(., z_k)$ and $I(., z_{k'})$. All these variograms are standardized to a unit sill

- All vectors of hard indicator data are complete: there are no missing values as could result from inequality constraints on the Z value.

When those two conditions are met, it is only necessary to infer one variogram $\gamma_Z(\mathbf{h})$ and only one kriging system has to be solved for all thresholds z_k (indeed, the data locations and the variogram are the same for all thresholds). This simplification is called *median indicator kriging*.

Ensuring the validity of the estimated cdf

Since kriging is a non-convex estimator, the cdf values estimated by kriging $Prob^*\big(Z(\mathbf{u}) < \mathrm{z_k}\big)$, $k = 1, \ldots, K$ estimated by indicator kriging may not satisfy the properties of a cdf F, that is:

$$\forall k \in \{1, ..., K\} \quad F(z_k) = Prob(Z(\mathbf{u}) < \mathrm{z_k}) \in [0, 1] \tag{2.2}$$

$$\forall z_k \geq z_{k'} \quad F(z_k) \geq F(z_{k'}) \tag{2.3}$$

If properties 2.2 and 2.3 are not verified, the program *Indicator Kriging* modifies the kriging values so as to ensure the validity of the ccdf. The corrections are performed in 3 steps.

1. Upward correction:
   - Loop through all the cutoffs z_1,..., z_K, starting with the lowest cutoff z_1.
   - At cutoff z_k, if the estimated probability $I^*(Z(\mathbf{u}), \mathrm{z_k})$ is not in $[I^*(Z(\mathbf{u}), \mathrm{z_{k-1}}), 1]$, reset it to the closest bound
2. Downward correction:
   - Loop through all the cutoffs z_1,..., z_K, starting with the highest cutoff z_K.
   - At cutoff z_k, if the estimated probability $I^*(Z(\mathbf{u}), \mathrm{z_k})$ is not in $[0, I^*(Z(\mathbf{u}), \mathrm{z_{k+1}})]$, reset it to the closest bound
3. The final corrected probability values are the average of the upward and downward corrections

Categorical Variables

Indicator Kriging can also be applied to categorical variables, i.e. variables that take a discrete, finite, number of values (also called classes or categories): $z(\mathbf{u}) \in \{0, ..., \mathrm{K} - 1\}$. The indicator variable for class k is then defined as:

$$I(\mathbf{u}, \mathrm{k}) = \begin{cases} 1 & \text{if } Z(\mathbf{u}) = \mathrm{k} \\ 0 & \text{otherwise} \end{cases}$$

and the probability $I^*(\mathbf{u}, \mathrm{k})$ of $Z(\mathbf{u})$ belonging to class k is estimated by simple kriging:

$$I^*(\mathbf{u}, \mathrm{k}) - \mathrm{E}\{\mathrm{I}(\mathbf{u}, \mathrm{k})\} = \sum_{\alpha=1}^{\mathrm{n}} \lambda_\alpha \Big(\mathrm{I}(\mathbf{u}_\alpha, \mathrm{k}) - \mathrm{E}\{\mathrm{I}(\mathbf{u}_\alpha, \mathrm{k})\} \Big)$$

In the case of categorical variables, the estimated probabilities must al be in $[0, 1]$ and verify:

$$\sum_{k=1}^{K} P\Big(Z(\mathbf{u} = \mathrm{k}) \mid (\mathrm{n}) \Big) = 1 \tag{2.4}$$

If not, they are corrected as follows:

1. If $I^*(Z(\mathbf{u}), \mathrm{z_k}) \notin [0, 1]$ reset it to the closest bound. If all the probability values are lesser or equal to 0, no correction is made and a warning is issued.

2. Standardize the values so that they sum-up to 1:

$$I^*_{corrected}(Z(\mathbf{u}), \mathrm{z_k}) = \frac{\mathrm{I^*(Z(\mathbf{u}), z_k)}}{\sum_{\mathrm{i=1}}^{\mathrm{K}} \mathrm{I^*(Z(\mathbf{u}), z_i)}}$$

Parameters description

`Hard_Data_Grid:` The grid containing the hard data

`Hard_Data_Property:` The name of the properties containing the hard data. If there are K indicators, K properties must be specified. Each property contains the indicator values $I(\mathbf{u}, \mathrm{k})$, $k = 1,\ldots,K$

`Categorical_Variable_Flag:` A flag indicating whether the variable to be kriged is categorical or continuous. If the flag is on (equal to 1), the variable is assumed categorical

`Marginal_Probabilities:` The marginal probabilities of each indicator. The probability values are separated by one or more space. The first probability corresponds to the probability of the first indicator and so on.

`Max_Conditioning_Data:` The maximum number of conditioning data to be used for each kriging

`Search_Ellipsoid:` The ranges and angles defining the search ellipsoid

`Median_Ik_Flag:` A flag indicating whether median IK should be performed

`Full_Ik_Flag:` A flag indicating whether full IK should be performed. If `Median_Ik_Flag` is on/off, `Full_Ik_Flag` is off/on

`Variogram_Median_Ik:` The variogram of the indicator variables used for median IK. If `Median_Ik_Flag` is off this parameter is ignored

`Variogram_Full_Ik:` The variograms of each indicator variable. If `Median_Ik_Flag` is on, this parameter is ignored

2.4 CoKriging

The aim of cokriging is to estimate a variable $Z_1(\mathbf{u})$ accounting for data on related variables $Z_2, ..., Z_{J+1}$. The cokriging estimator is given by:

$$Z_1^*(\mathbf{u}) - \mathrm{m}_1(\mathbf{u}) = \sum_{\alpha} \lambda_\alpha \Big[\mathrm{Z}_1(\mathbf{u}_\alpha) - \mathrm{m}_1(\mathbf{u}_\alpha)\Big] + \sum_{\mathrm{j}=2}^{\mathrm{J}+1} \sum_{\beta_\mathrm{j}}^{\mathrm{N_j}} \lambda_{\beta_\mathrm{j}} \Big[\mathrm{Z_j}(\mathbf{u}_{\beta_\mathrm{j}}) - \mathrm{m_j}(\mathbf{u}_{\beta_\mathrm{j}})\Big] \quad (2.5)$$

Algorithm *CoKriging* allows to distinguish between the two cases of simple and ordinary cokriging:

- the means $m_1(\mathbf{u}), ..., \mathrm{m}_{\mathrm{J}+1}(\mathbf{u})$ are known and constant: simple cokriging
- the means are locally constant but unknown. The weights λ_α and λ_{β_j} must then satisfy:

$$\sum_{\alpha} \lambda_\alpha = 1 \quad (2.6)$$

$$\sum_{\beta_j}^{N_j} \lambda_{\beta_j} = 0 \quad \forall j \in \{1, ..., J\} \quad (2.7)$$

Solving the cokriging system calls for the inference and modeling of multiple variograms: the variogram of each variable and all cross-variograms between any two variables. In order to alleviate the burden of modeling all these variograms, two models have been proposed: the Markov Model 1 and the Markov Model 2 [Almeida and Journel, 1994; Goovaerts, 1997b]

For text clarity, we will only consider the case of a single secondary variable ($J = 1$).

Markov Model 1

The Markov Model 1 (MM1) considers the following Markov-type screening hypothesis:

$$E\Big(Z_2(\mathbf{u}) \mid \mathrm{Z}_1(\mathbf{u}), \mathrm{Z}_1(\mathbf{u}+\mathbf{h})\Big) = \mathrm{E}\Big(\mathrm{Z}_2(\mathbf{u}) \mid \mathrm{Z}_1(\mathbf{u})\Big) \tag{2.8}$$

i.e. the dependence of the secondary variable on the primary is limited to the co-located primary variable.

The cross-variogram (or cross-covariance) is then proportional to the auto-variogram of the primary variable [Almeida and Journel, 1994; Goovaerts, 1997b]:

$$C_{12}(\mathbf{h}) = \frac{\mathrm{C}_{12}(\mathbf{0})}{\mathrm{C}_{11}(\mathbf{0})}\mathrm{C}_{11}(\mathbf{h}) \tag{2.9}$$

Where C_{12} is the covariance between Z_1 and Z_2 and C_{11} is the covariance of Z_1. Solving the cokriging system under the MM1 model only calls for knowledge of C_{11}, hence the inference and modeling effort is the same as for univariate kriging.

Although very congenial the MM1 model should not be used when the support of the secondary variable Z_2 is larger than the one of Z_1, lest the variance of Z_1 would be underestimated. It is better to use the Markov Model 2 in that case.

Markov Model 2

The Markov Model 2 (MM2) was developed for the case where the volume support of the secondary variable is larger than that of the primary variable [Journel, 1999]. This is often the case with remote sensing and seismic-related data. The Markov-type hypothesis is now:

$$E\Big(Z_1(\mathbf{u}) \mid \mathrm{Z}_2(\mathbf{u}), \mathrm{Z}_2(\mathbf{u}+\mathbf{h})\Big) = \mathrm{E}\Big(\mathrm{Z}_1(\mathbf{u}) \mid \mathrm{Z}_2(\mathbf{u})\Big) \tag{2.10}$$

i.e. the dependence of the primary variable on the secondary is limited to the co-located secondary variable.

The cross-variogram is now proportional to the variogram of the secondary variable:

$$C_{12}(\mathbf{h}) = \frac{\mathrm{C}_{12}(\mathbf{0})}{\mathrm{C}_{11}(\mathbf{0})}\mathrm{C}_{22}(\mathbf{h}) \tag{2.11}$$

In order for all three covariances C_{11}, C_{12} and C_{22} to be consistent, Journel (1999) proposed to model C_{11} as a linear combination of C_{22} and any permissible residual covariance C_R. Expressed in term of correlograms ($C_{11}(\mathbf{h}) = \mathrm{C}_{11}(\mathbf{0})\rho_{11}(\mathbf{h})$), this is written:

$$\rho_{11}(\mathbf{h}) = \rho_{12}^2\ \rho_{22}(\mathbf{h}) + (1 - \rho_{12}^2)\ \rho_{\mathrm{R}}(\mathbf{h}) \tag{2.12}$$

Parameters description

`Primary_Harddata_Grid:` The grid containing the primary hard data

`Primary_Variable:` The name of the property containing the primary hard data

`Assign_Hard_Data:` A flag specifying whether the hard data should be re-located on the simulation grid.

`Secondary_Harddata_Grid:` The grid containing the secondary hard data

`Secondary_Variable:` The name of the property containing the secondary hard data

`Kriging_Type:` Possible types of cokriging include: Simple Kriging (SK) and Ordinary Kriging (OK). Note that selecting OK while retaining only a single secondary datum (e.g. when doing co-located cokriging) amounts to completely ignoring the secondary information : from Eq. (2.7), the weight associated to the single secondary datum will be 0.

`SK_Means:` The mean of the primary and secondary variables. If the selected kriging type is Ordinary Kriging, this parameter is ignored

`Cokriging_Type:` The cokriging option. Available options are: Full Cokriging, Co-located Cokriging with Markov Model 1 (MM1), and Co-located Cokriging with Markov Model 2 (MM2)

`Max_Conditioning_Data:` The maximum number of primary conditioning data to be used for each cokriging

`Search_Ellipsoid_1:` The ranges and angles defining the search ellipsoid used to find the primary conditioning data

`Variogram_C11:` The variogram model for the primary variable

`Max_Conditioning_Data_2:` The maximum number of secondary conditioning data used for each cokriging. This parameter is only required if `Cokriging_Type` is set to Full Cokriging

`Search_Ellipsoid_2:` The ranges and angles defining the search ellipsoid used to find the secondary conditioning data. This parameter is only required if `Cokriging_Type` is set to Full Cokriging

`Variogram_C12:` The cross-variogram between the primary and secondary variable. This parameter is only required if `Cokriging_Type` is set to Full Cokriging

`Variogram_C22:` The variogram of the secondary variable. This parameter is only required if `Cokriging_Type` is set to Full Cokriging

`Correl_Z1Z2:` The coefficient of correlation between the primary and secondary variable. This parameter is only required if `Cokriging_Type` is set to MM1

`MM2_Correl_Z1Z2:` The coefficient of correlation between the primary and secondary variable. This parameter is only required if `Cokriging_Type` is set to MM2

`MM2_Variogram_C22:` The variogram of the secondary variable. This parameter is only required if `Cokriging_Type` is set to MM2. Note that the variogram of the secondary variable and the variogram of the primary variable must verify the condition (2.12)

Section 3

Simulation Algorithms

Stochastic simulations provides a depiction and a measure of uncertainty of the spatial variability of a phenomenon. This is done by generating multiple realizations of the stochastic process modeling the spatial distribution under study. Several simulation techniques have been developed, which can be classified into four categories: sequential simulation, p-field simulation, object simulation and optimization-based techniques.

All the simulation tools implemented in **GEMS** belong to the sequential simulation category. Sequential simulation is a broad class of algorithms which can be used to solve a very large spectrum of problems. It relies on the property that the joint spatial conditional distribution of a variable $Z(\mathbf{u})$ at several (N) locations in space can be decomposed into the following product:

$$\begin{aligned} F\Big(\mathbf{u}_1,\ldots,\mathbf{u_N};\mathrm{z}_1,\ldots,\mathrm{z}_\mathrm{N}\mid(\mathrm{n})\Big) = {} & F(\mathbf{u}_\mathrm{N};\mathrm{z}_\mathrm{N}\mid(\mathrm{n}+\mathrm{N}-1))* \\ & F(\mathbf{u}_{\mathrm{N}-1};\mathrm{z}_{\mathrm{N}-1}\mid(\mathrm{n}+\mathrm{N}-2))* \\ & \vdots \\ & F(\mathbf{u}_2;\mathrm{z}_2\mid(\mathrm{n}+1))*\mathrm{F}(\mathbf{u}_1;\mathrm{z}_1\mid(\mathrm{n})) \end{aligned} \tag{3.1}$$

where $F(\mathbf{u}_\mathrm{j};\mathrm{z}_\mathrm{j}\mid(\mathrm{n}+\mathrm{j}))$ is the probability that any single variable $Z(\mathbf{u}_\mathrm{j})\leq \mathrm{z}_\mathrm{j}$ given $n+j$ conditioning data $(n+j)$, $j=1,\ldots,N$. Hence to sample from the joint distribution $F\Big(\mathbf{u}_1,\ldots,\mathbf{u_N};\mathrm{z}_1,\ldots,\mathrm{z}_\mathrm{N}\mid(\mathrm{n})\Big)$ one could sequentially sample from N single-point conditional distributions $F(\mathbf{u}_\mathrm{j};\mathrm{z}_\mathrm{j}\mid(\mathrm{n}+\mathrm{j}))$, $j=1,\ldots,N$, each sampled value becoming a conditioning datum for the next ccdf.

Notice that in Eq. 3.1 each ccdf $F(\mathbf{u}_\mathrm{j}; \mathrm{z}_\mathrm{j} \mid (\mathrm{n}+\mathrm{j}))$ involves all $(n+j)$ data. Conditioning to all $(n+j)$ data can be daunting and it is a common assumption that $F(\mathbf{u}_\mathrm{j}; \mathrm{z}_\mathrm{j} \mid (\mathrm{n}+\mathrm{j})) = \mathrm{F}(\mathbf{u}_\mathrm{j}; \mathrm{z}_\mathrm{j} \mid (\mathrm{n}'+\mathrm{j}'))$, where $(n'+j')$ is a subset of the $(n+j)$ data found in a neighborhood of $\mathbf{u}_\mathrm{j}$. This Markov-type assumption is often justified by the *screening effect* of remote data by the closest $(n'+j')$ data retained.

Another point worth noticing is that in order to actually sample from $F\big(\mathbf{u}_1, \ldots, \mathbf{u}_\mathbf{N}; \mathrm{z}_1, \ldots, \mathrm{z}_\mathrm{N} \mid (\mathrm{n})\big)$ all the conditional cumulative distribution function's in decomposition 3.1 must be exactly known. That is generally not the case.

3.1 Common Simulation Parameters

The following parameters are common to all simulation algorithms:

Parameters description

`Grid_Name:` The grid on which the simulation algorithm is to be performed

`Property_Name_Sim:` Each realization is stored as a new property attached to the object specified by `Grid_Name`. The properties are called `Property_Name_Sim` followed by a unique index

`Nb_Realizations:` The number of realizations to be generated

`Seed:` The seed of the random number generator used for the stochastic simulation

3.2 SGSIM

Let $Z(\mathbf{u})$ be a multivariate Gaussian random function with 0 mean, unit variance, and a given covariance model. Realizations of Z, conditioned to data (n) can be generated by **Algorithm 1**.

It can be shown [Goovaerts, 1997a] that the resulting realizations will reproduce both the first and second moments (covariance) of Z. Note that the theory requires the variance of each ccdf be estimated by the simple kriging variance. Algorithm *SGSIM* can account

Algorithm 1 Sequential Gaussian Simulation

1. Define a random path visiting each node of the grid once
2. At each node **u**, the local conditional cumulative distribution function is Gaussian, its mean is estimated by simple kriging and its variance is equal to the simple kriging variance. The conditioning data consist of both neighboring original data (n) and previously simulated values
3. Draw a value from that Gaussian ccdf and add the simulated value to the data set
4. Proceed to the next node in the path and repeat the two previous steps until all nodes have been visited

for non-stationary behaviors by using other types of kriging (see 2.2) to estimate the mean and variance of the Gaussian cdf. In that case however, variogram reproduction is not theoretically ensured.

Histogram Transformation

The sequential Gaussian simulation algorithm as described above assumes that the variable is Gaussian. If that is not the case, it is possible to transform the marginal distribution of the variable into a Gaussian distribution and work on the transformed variable. The transformation of variable $Z(\mathbf{u})$ with cdf F_Z into a standard normal variable $Y(\mathbf{u})$ with cumulative distribution function G is written:

$$Y(\mathbf{u}) = \mathrm{G}^{-1}\Big(\mathrm{F}\Big(\mathrm{Z}(\mathbf{u})\Big)\Big) \tag{3.2}$$

This transformation does not ensure that Y is multivariate Gaussian, only its histogram is. One should check that the multivariate (or at least bi-variate) Gaussian hypothesis holds for Y before performing Gaussian simulation. If the hypothesis is not appropriate, other algorithms that do not require Gaussianity, e.g. sequential indicator simulation (*SISIM*) should be considered.

The sequential Gaussian simulation algorithm proceeds as follows:

1. Transform Z into a Gaussian variable according to Eq. (3.2)

2. Simulate Y as in **Algorithm 1**

3. "Back-transform" the simulated values $y_1, ..., y_N$ into $z_1, ..., z_N$:

$$z_i = F^{-1}\Big(G(y_i)\Big) \quad i = 1,\ldots,N \tag{3.3}$$

In *SGSIM* the distribution function F is infered from a set of values $\zeta_1 \leq \ldots \leq \zeta_L$ which can either be the hard data or values read from a file. F is built such that: $F(\zeta_1) = \frac{1}{L}$ and $F(\zeta_L) = 1 - \frac{1}{L}$. The back-transformation step of *SGSIM* requires in addition the modeling of the tails of the target distribution $F(z)$, for values outside the data interval: $z < \zeta_1$ or $z > \zeta_L$.

Two options are available in *SGSIM*:

- Z is bounded: $F(Z < z_{min}) = F(Z > z_{max}) = 0$. The lower tail of F is then modeled by:
$$\frac{F(\zeta_1) - F(z)}{F(\zeta_1)} = \Big(\frac{\zeta_1 - z}{\zeta_1 - z_{min}}\Big)^3 \quad \forall z \in]z_{min}, \zeta_1[\tag{3.4}$$
and the upper tail:
$$\frac{F(z) - F(\zeta_L)}{1 - F(\zeta_L)} = \Big(\frac{z - \zeta_L}{z_{max} - \zeta_1}\Big)^{1/3} \quad \forall z \in]\zeta_L, z_{max}[\tag{3.5}$$

- Z is not bounded: the lower tail is then given by:
$$F(z) = F(\zeta_1)\ \exp\big(-(z - \zeta_1)^2\big) \quad \forall z < \zeta_1 \tag{3.6}$$
and the uppert tail:
$$\frac{1 - F(z)}{1 - F(\zeta_L)} = \frac{\zeta_L}{z} \quad \forall z > \zeta_L \tag{3.7}$$

Example:
Algorithm *SGSIM* is used to simulate the shale porosity in the second layer of Stanford V. The histogram of the shale porosity data is shown in Fig 3.2. It is used as the reference histogram in the normal score transformation of the data (see the discussion about histogram transformation). Since the data are too sparse to allow a reliable inference of the variogram,

the variogram is arbitrarily modeled by an isotropic spherical variogram, of range 40. Different views of a realization are show in Fig. 3.3. The parameter file corresponding to the run is reproduced on Fig. 3.2.

Fig. 3.2 and 3.2 show that both the target distribution and variogram are correctly reproduced.

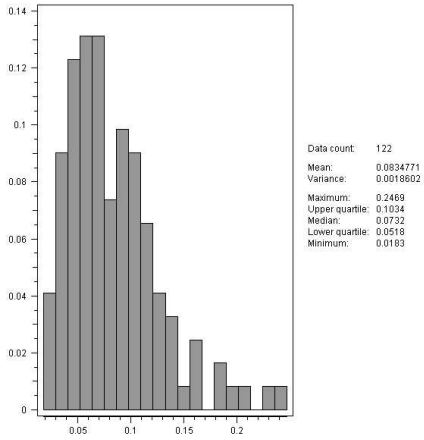
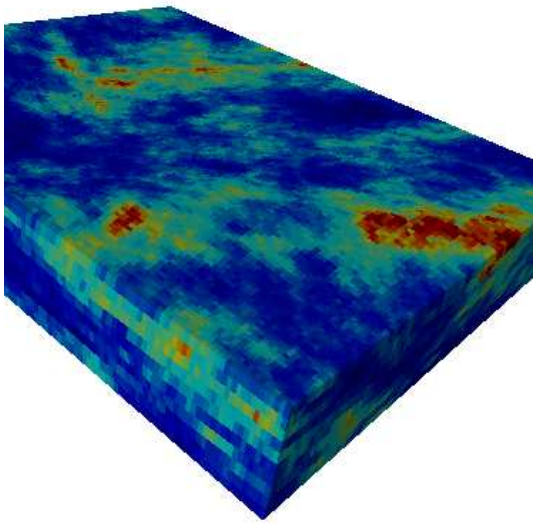


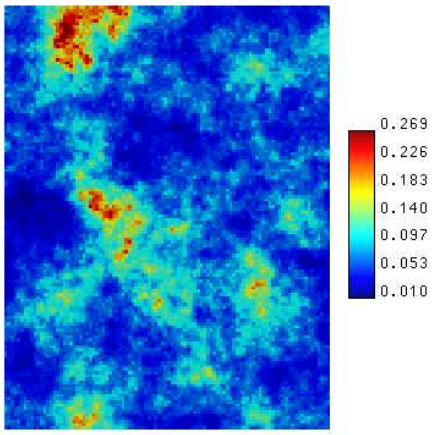
Figure 3.1: Histogram of shale porosity

```
<parameters>  <algorithm name="sgsim" />
  <Grid_Name  value="layer 2"  />
  <Property_Name  value="sgsim_shale_porosity" />

  <Nb_Realizations  value="5" />
  <Seed  value="14071789" />

  <Hard_Data  grid="layer 2 shale data"   property="porosity"  />
  <Assign_Hard_Data  value="1"  />
  <Use_Target_Histogram  value="1"  />
  <Target_Hist_From_Harddata  value="1"  />
  <Target_Hist_From_File  value="0"  />
  <Use_Min_Max  value="1"  />
  <Target_Hist_Min  value="0.01" />
  <Target_Hist_Max  value="0.27" />

  <Kriging_Type  value="Simple Kriging (SK)"  />

  <Max_Conditioning_Data  value="15" />
  <Search_Ellipsoid  value="70 70 10
                            0 0 0 " />

  <Variogram  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="30"  medium="30"  min="5"   />
      <angles x="0"  y="0"  z="0"   />
    </structure_1>
  </Variogram>
</parameters>
```

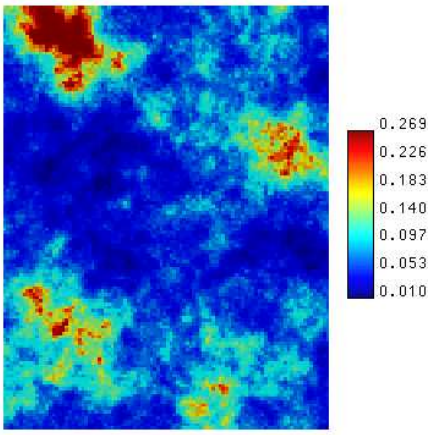
Figure 3.2: Sequential Gaussian simulation parameters



(a) 3D view



(b) Cross-section 1



(c) Cross-section 2

Figure 3.3: shale porosity simulated with *SGSIM*

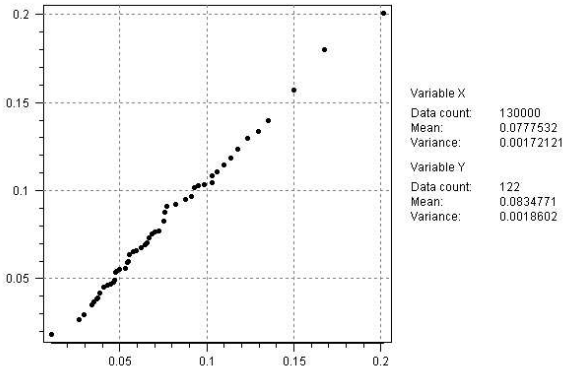


Figure 3.4: Reproduction of the target histogram

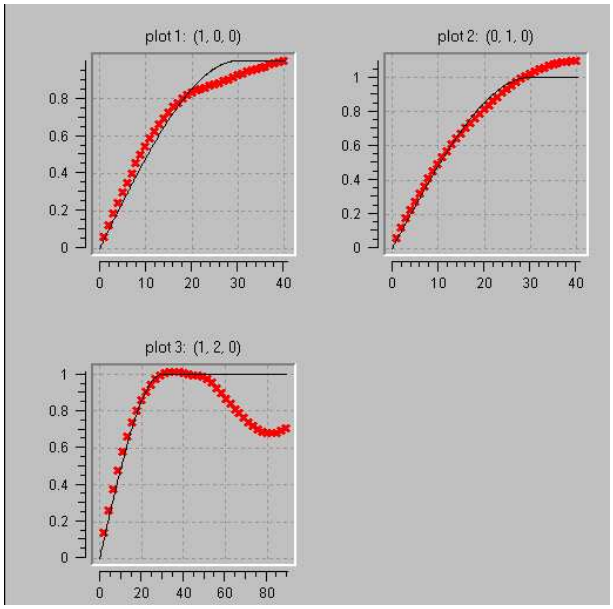


Figure 3.5: Reproduction of the input variogram. The continuous black line is the input variogram, the red crosses the variogram of the SGSIM realization

Parameters description

`Hard_Data:` The grid and the name of the property containing the hard data

`Assign_Hard_Data:` A flag specifying whether the hard data should be re-located on the simulation grid.

`Use_Target_Histogram:` A flag indicating whether to perform steps 1 and 3 of **Algorithm 1**. If the flag is set, distribution F (use in relations (3.2) and (3.3)) must be specified. The distribution can either be infered from the hard data, if any, or defined explicitely (see parameters `Target_Hist_From_Harddata` and `Target_Hist_From_File`). Setting this flag is necessary if the hard data do not follow a standard normal distribution, as dicussed in Section 3.2

`Target_Hist_From_Harddata:` If this flag is on (equal to 1), the target histogram corresponding to cdf F will be infered from the hard data

`Target_Hist_From_File:` If this flag is on, the histogram is read from a file. The file name is specified by `Target_Hist_File`. Note that `Target_Hist_From_Harddata` and `Target_Hist_From_File` are mutually exclusive : both can not be set to 1 at the same time.

`Target_Hist_File:` The name of the file containing the histogram description. This is an ASCII file containing values drawn from the target distribution. The values can be separated by spaces, tabulations of return carriage. The target distribution is computed from that list of values. Any number of values can be specified in the file. This parameter is ignored if `Target_Hist_From_File` is set to 0

`Use_Min_Max:` A flag indicating whether variable Z is bounded: $z_{min} \leq Z(\mathbf{u}) \leq \mathrm{z_{max}}\ \forall \mathbf{u}$.

`Target_Hist_Min:` The minimum of the target distribution. This parameter is ignored if `Use_Min_Max` is set to 0

`Target_Hist_Max:` The maximum of the target distribution. This parameter is ignored if `Use_Min_Max` is set to 0

`Kriging_Type:` The possible types of kriging are: Simple Kriging (SK), Ordinary Kriging (OK), Kriging with Trend (KT) and Simple Kriging with Localy Varying Mean (LVM). Because of Gaussian simulation theory, simple kriging should be the prefered option unless deemed inappropriate.

`Trend:` The trend components. The possible components are, with $\mathbf{u} = (\mathrm{x}, \mathrm{y}, \mathrm{z})$ being the coordinates vector:

$t_1(\mathbf{u}) = \mathrm{x}$, $t_2(\mathbf{u}) = \mathrm{y}$, $t_3(\mathbf{u}) = \mathrm{z}$, $t_4(\mathbf{u}) = \mathrm{x.y}$, $t_5(\mathbf{u}) = \mathrm{x.z}$, $t_6(\mathbf{u}) = \mathrm{y.z}$, $t_7(\mathbf{u}) = \mathrm{x}^2$, $t_8(\mathbf{u}) = \mathrm{y}^2$, $t_9(\mathbf{u}) = \mathrm{z}^2$.

The trend is coded by a string of 9 flags, for example string "`0 1 0 0 0 1 0 1 0''` corresponds to a trend with only components t_2, t_6 and t_8, i.e.: $T(\mathbf{u}) = \alpha\ \mathrm{x} + \beta\ \mathrm{yz} + \delta\ \mathrm{y}^2$. Each flag is separated by a space.

`Local_Mean_Property:` The property of the simulation grid containing the non-stationary mean. A mean value must be available at each location to be simulated.

`Max_Conditioning_Data:` The maximum number of conditioning data used to infer the local conditional distribution at each simulation node

`Search_Ellipsoid:` The ranges and angles defining the search ellipsoid

`Variogram:` The variogram model for the variable to be simulated

3.3 SISIM

The algorithm *SISIM* relies on indicator kriging (see Section 2.3) to infer the conditional cumulative distribution function s $F_Z(\mathbf{u}_{\mathrm{k}+1} \mid (\mathrm{n} + \mathrm{k}))$, $k = 1, \ldots, N-1$ of Eq. (3.1).

Let $Z(\mathbf{u})$ be a continuous variable. *SISIM* implements **Algorithm 2**.

Step 2(c) of **Algorithm 2** calls for the solution of K kriging systems. Computations can actually be alleviated by using the same indicator variogram for all thresholds (see discussion on median indicator kriging in section 2.3).

Algorithm 2 SISIM - continuous variable

1. Choose a discretization of the range $\mathcal{D}$ of $Z(\mathbf{u})$: $z_1, \ldots, z_K$
2. Define a path visiting all locations to be simulated
3. For each location **u** along the path:
   (a) Retrieve the neighboring conditioning data of location **u**: $z(\mathbf{u}_\alpha)$, $\alpha = 1, \ldots, N$
   (b) Turn each conditioning datum $z(\mathbf{u}_\alpha)$ into a vector of indicator values: $\mathbf{v}(\mathbf{u}_\alpha) = \Big[\mathrm{i}\big(\mathrm{z}(\mathbf{u}_\alpha), \mathrm{z}_1\big), \ldots, \mathrm{i}\big(\mathrm{z}(\mathbf{u}_\alpha), \mathrm{z}_\mathrm{K}\big)\Big]$. The indicator variable is defined as:

$$i(\mathbf{u}, \mathrm{z_k}) = \begin{cases} 1 & \text{if } z(\mathbf{u}) \leq \mathrm{z_k} \\ 0 & \text{otherwise} \end{cases}$$

   (c) Estimate the indicator random variable $I(\mathbf{u}, \mathrm{z_k})$ for each of the K cutoffs by solving a kriging system. The simple kriging estimator is given by:

$$I^*(\mathbf{u}, \mathrm{z_k}) - \mathrm{E}\{\mathrm{I}(\mathbf{u}, \mathrm{z_k})\} = \sum_{\alpha=1}^{\mathrm{N}} \lambda_\alpha \Big(\mathrm{I}(\mathbf{u}_\alpha, \mathrm{z_k}) - \mathrm{E}\{\mathrm{I}(\mathbf{u}_\alpha, \mathrm{z_k})\}\Big)$$

   (d) The estimated values $i^*(\mathbf{u}, \mathrm{z_k}) = \mathrm{Prob}^*(\mathrm{Z}(\mathbf{u}) \leq \mathrm{z_k})$, after correction of order-relation problems (see Section 2.3), define an estimate of the cumulative distribution function $F_{Z(\mathbf{u})}$ of the variable $Z(\mathbf{u})$. Draw a value from that ccdf and assign it as a datum at location **u**.
   (e) Loop until all locations are visited
4. Repeat the previous steps to generate another simulated realization

Sampling the estimated distribution function

At each location to be simulated, the ccdf $F_{Z(\mathbf{u})}$ is estimated at values $z_1, \ldots, z_K$. However sampling from $F_{Z(\mathbf{u})}$ as described in step 3(e) of **Algorithm 2** requires the knowledge of $F_{Z(\mathbf{u})}(z)$ for all $z \in \mathcal{D}$ (see **Algorithm 2**). In *SISIM* $F_{Z(\mathbf{u})}$ is interpolated as follows:

$$F^*_{Z(\mathbf{u})}(z) = \begin{cases} F^*_{Z(\mathbf{u})}(z_1)\; e^{-(z_1-z)^2} & \text{if } z \leq z_1 \\ F^*_{Z(\mathbf{u})}(z_1) + \frac{z-z_k}{z_{k+1}-zk}\big(F^*_{Z(\mathbf{u})}(z_2) - F^*_{Z(\mathbf{u})}(z_1)\big) & \text{if } z_k \leq z \leq zk+1 \\ 1 - \big(1 - F^*_{Z(\mathbf{u})}(z_K)\big)\big(\frac{z_K}{z}\big)^{\omega} & \text{if } z_{k+1} \leq z, \quad \omega > 0 \end{cases} \tag{3.8}$$

where $F^*_{Z(\mathbf{u})}(z_k) = i^*(\mathbf{u}, \mathrm{z_k})$ is estimated by indicator kriging.

Categorical Variables

If $Z(\mathbf{u})$ is a categorical variable which only takes the integer values $\{1, \ldots, K\}$, **Algorithm 2** is modified as described in **Algorithm 3**.

Implementation limitations

Algorithm *SISIM* does not allow for inequality constraints or soft data. If such data must be accounted for, use algorithm *COSISIM* instead, see section 4.

Example:

Porosity in the second layer of Stanford V is bi-modal (see Fig 1.16): it is higher in the sand facies than in shale. Its spatial distribution is also variable: the high porosity region has a 45^o azimuth anisotropy, while the low porosity is isotropic. Full indicator simulation is used to simulate such a complex field. Five thresholds are selected such that each of the two modes of the porosity distribution is described by the same number of thresholds. The cut-offs $z_1, \ldots, z_5$ retained and the corresponding probabilities $P(Z \leq z_1), \ldots, P(Z \leq z_5)$ are:

threshold z_i	0.06	0.15	0.23	0.27	0.3
$P(Z \leq z_i)$	0.22	0.54	0.63	0.69	0.9

The variograms corresponding to the three first thresholds are spherical isotropic models, of range 40, while the last two variograms have a 45^o azimuth anisotropy and an anisotropy ratio of 3. The parameter file for the simulation is reproduction in Fig 3.6.

Algorithm 3 SISIM - categorical variable

1. Define a path visiting all locations to be simulated
2. For each location **u** along the path:
   (a) Retrieve the neighboring categorical conditioning data: $z(\mathbf{u}_\alpha)$, $\alpha = 1, \ldots, N$
   (b) Turn each datum $z(\mathbf{u}_\alpha)$ into a vector of indicator data values: $\mathbf{v}(\mathbf{u}_\alpha) = \Big[\mathrm{i}\big(\mathrm{z}(\mathbf{u}_\alpha), \mathrm{z}_1\big), \ldots, \mathrm{i}\big(\mathrm{z}(\mathbf{u}_\alpha), \mathrm{z}_\mathrm{K}\big)\Big]$. The indicator variable is defined by:

$$i(\mathbf{u}, \mathrm{k}) = \begin{cases} 1 & \text{if } Z(\mathbf{u}) = \mathrm{k} \\ 0 & \text{otherwise} \end{cases}$$

   (c) Estimate the indicator random variable $I(\mathbf{u}, \mathrm{k})$ for each of the K categories by solving a kriging system. The simple kriging estimator is given by:

$$I^*(\mathbf{u}, \mathrm{k}) - \mathrm{E}\{\mathrm{I}(\mathbf{u}, \mathrm{k})\} = \sum_{\alpha=1}^{\mathrm{N}} \lambda_\alpha \Big(\mathrm{I}(\mathbf{u}_\alpha, \mathrm{k}) - \mathrm{E}\{\mathrm{I}(\mathbf{u}_\alpha, \mathrm{k})\}\Big)$$

   (d) The estimated values $i^*(\mathbf{u}, \mathrm{k}) = \mathrm{prob}^*(\mathrm{Z}(\mathbf{u}) = \mathrm{k})$, after correction of order-relation problems (see 2.3), define an estimate of the discrete probability density function (pdf) $f_{Z(\mathbf{u})}$ of the categorical variable $Z(\mathbf{u})$. Draw a realization from f and assign it as a datum at location **u**.
   (e) Loop until all locations are visited
3. Repeat the previous steps to generate another realization

Different views of a sample realization are shown on Fig 3.7.

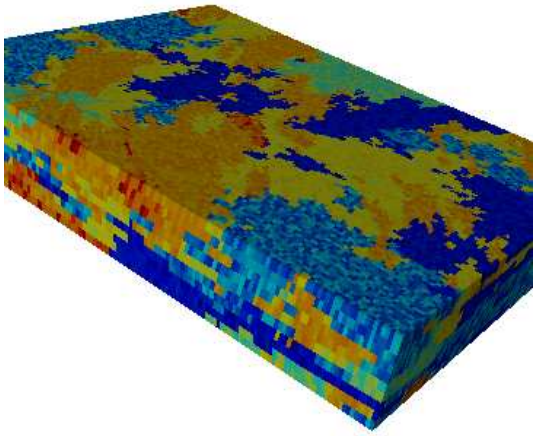
Example:

As noted in the previous example, the complexity of the porosity field of Stanford V layer 2, comes from its non-stationarity: porosity is high in the sand regions and low in the shale regions. Full indicator simulation provides the necessary flexibility to simulate such a complex field. Another possible approach is to first separate the field into two stationary regions, channel sand and shale, and then simulate porosity in each region independently.

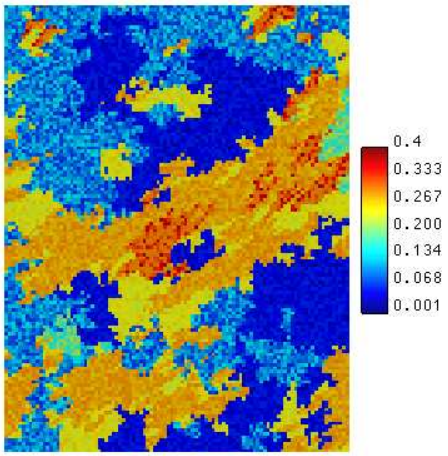
In this example, algorithm *SISIM* is used to simulate the two sand and shale facies. Full indicator simulation is chosen to simulate sand bodies preferentially oriented along a 45^o direction and isotropic shale bodies. A long range is used for the sand variogram to mimic the high connectivity of the actual channels of layer 2 (see Fig. 1.15). The input parameters are reproduced in Fig 3.8. Different views of a sample realization are shown in Fig 3.9. It can be noticed that while the connectivity of the simulated sand channels is high, the sinuosity of the reference channels is not reproduced.

```
<parameters>  <algorithm name="sisim" />
  <Grid_Name  value="layer 2"  />
  <Property_Name  value="sisim" />
  <Nb_Realizations  value="5" />
  <Seed  value="14071789" />
  <Hard_Data_Grid  value="layer 2 well data"  />
  <Hard_Data_Property  value="porosity"  />
  <Categorical_Variable_Flag  value="0"  />
  <Nb_Indicators  value="5" />
  <Thresholds  value="0.06   0.15  0.23  0.27  0.3" />
  <Marginal_Probabilities  value="0.22  0.54  0.63  0.69  0.9" />
  <Use_Min_Max  value="1"  />
  <Cdf_Min  value="0" />
  <Cdf_Max  value="0.4" />
  <Max_Conditioning_Data  value="15" />
  <Search_Ellipsoid  value="70 70 6
                            0 0 0 " />
  <Full_Ik_Flag  value="1"  />
  <Variogram_Full_Ik  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="40"  medium="40"  min="5"   />
      <angles x="0"  y="0"  z="0"   />
    </structure_1>
  </Variogram_Full_Ik>
  <Variogram_Full_Ik_2  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="40"  medium="40"  min="5"   />
      <angles x="0"  y="0"  z="0"   />
    </structure_1>
  </Variogram_Full_Ik_2>
  <Variogram_Full_Ik_3  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="40"  medium="40"  min="5"   />
      <angles x="0"  y="0"  z="0"   />
    </structure_1>
  </Variogram_Full_Ik_3>
  <Variogram_Full_Ik_4  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="45"  medium="15"  min="5"   />
      <angles x="30"  y="0"  z="0"   />
    </structure_1>
  </Variogram_Full_Ik_4>
  <Variogram_Full_Ik_5  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="45"  medium="15"  min="5"   />
      <angles x="45"  y="0"  z="0"   />
    </structure_1>
  </Variogram_Full_Ik_5>
</parameters>
```

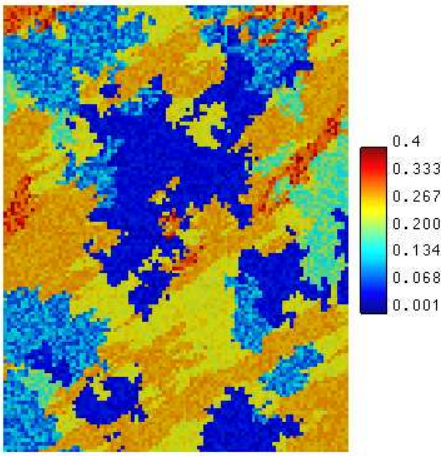
Figure 3.6: Sequential indicator simulation parameters



(a) 3D view



(b) Cross-section 1

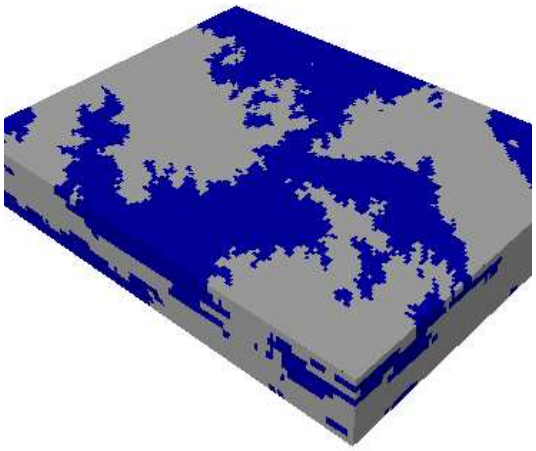


(c) Cross-section 2

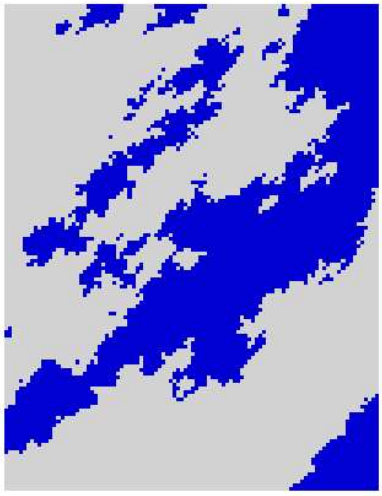
Figure 3.7: Porosity simulated with full indicator simulation

```
<parameters>  <algorithm name="sisim" />
  <Grid_Name  value="layer 2"  />
  <Property_Name  value="sisim" />

  <Nb_Realizations  value="5" />
  <Seed  value="14071789" />

  <Hard_Data_Grid  value="well data 2 facies"  />
  <Hard_Data_Property  value="facies     "  />

  <Categorical_Variable_Flag  value="1"  />
  <Nb_Indicators  value="2" />
  <Marginal_Probabilities  value="0.63  0.37" />
  <Cdf_Min  value="0" />
  <Cdf_Max  value="0.4" />

  <Max_Conditioning_Data  value="15" />
  <Search_Ellipsoid  value="70 70 6
                            0 0 0 " />

  <Median_Ik_Flag  value="0"  />
  <Full_Ik_Flag  value="1"  />

  <Variogram_Full_Ik  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="40"  medium="40"  min="5"   />
      <angles x="0"  y="0"  z="0"   />
    </structure_1>
  </Variogram_Full_Ik>
  <Variogram_Full_Ik_2  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="60"  medium="20"  min="5"   />
      <angles x="45"  y="0"  z="0"   />
    </structure_1>
  </Variogram_Full_Ik_2>
</parameters>
```

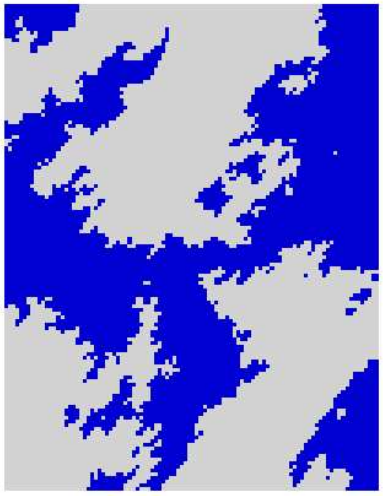
Figure 3.8: Sequential indicator simulation parameters - Facies simulation



(a) 3D view



(b) Cross-section 1



(c) Cross-section 2

Figure 3.9: Sand (in blue) and shale (in white) facies simulated with full indicator simulation

Parameters description

`Hard_Data_Grid:` The grid containing the hard data property

`Hard_Data_Property:` The name of the property containing the hard data

`Categorical_Variable_Flag:` A flag indicating whether the variable to be simulated is categorical or continuous. If the flag is on (i.e. equal to 1), the variable is categorical

`Thresholds:` The threshold values $z_1, \ldots, z_K$ defining each indicator variable. This is only needed for continuous variables. The threshold values are separated by spaces. This parameter is ignored if `Categorical_Variable_Flag` is set to 1

`Marginal_Probabilities:` The marginal probabilities of each indicator. The probability values are separated by spaces. In the case of a continuous variable (flag `Categorical_Variable_Flag` set to 0) the marginal probability values $p_1, \ldots, p_K$ must define a valid cumulative distribution function, that is:

$$p_k \in [0, 1] \qquad \forall\, k$$
$$p_k \leq p_{k'}, \quad \forall\, k \leq k'$$

If the variable is categorical (flag `Categorical_Variable_Flag` set to 1), $p_1, \ldots, p_K$ must define a valid pdf, that is:

$$p_k \in [0, 1] \quad \forall\, k$$
$$\sum_{k=1}^{K} p_k = 1$$

`Max_Conditioning_Data:` The maximum number of conditioning data used for each kriging

`Search_Ellipsoid:` The ranges and angles defining the search ellipsoid

`Median_Ik_Flag:` A flag indicating whether median IK should be performed

`Full_Ik_Flag:` A flag indicating whether full IK should be performed. Note that `Median_Ik_Flag` and `Full_Ik_Flag` are mutually exclusive

`Variogram_Median_Ik:` The single indicator variogram model used for median IK. If `Median_Ik_Flag` is off this parameter is ignored

`Variogram_Full_Ik:` The indicator variogram models (one for each indicator). If `Median_Ik_Flag` is on this parameter is ignored

3.4 COSGSIM

COSGSIM allows to simulate a Gaussian variable accounting for secondary information. Let $Z_1(\mathbf{u})$ and $Z_2(\mathbf{u})$ be two correlated random variables. $Z_1(\mathbf{u})$ is called the primary variable, and $Z_2(\mathbf{u})$ the secondary variable. The implementation of the *COSGSIM* simulation of the primary variable Z_1 conditioned to both primary and secondary data is described in **Algorithm 4**.

Accounting for non stationarity

Reproduction of the covariance of Z_1 is ensured in theory only if simple cokriging is used in step 2(b) of **Algorithm 4**. Simple cokriging requires that the expected values of Z_1 and Z_2 are known. If the means are not known, *COSGSIM* allows the use of ordinary cokriging in step 2(b) of **Algorithm 4**. However, the reproduction of the covariance of Z_1 is not guaranteed anymore.

The Markov Models for coregionalization

As discussed in section 2.4, it is possible to simplify the inference and modeling of the three covariances $Cov\big(Z_1(\mathbf{u}), \mathrm{Z}_1(\mathbf{u}+\mathbf{h})\big)$, $Cov\big(Z_1(\mathbf{u}), \mathrm{Z}_2(\mathbf{u}+\mathbf{h})\big)$, $Cov\big(Z_2(\mathbf{u}), \mathrm{Z}_2(\mathbf{u}+\mathbf{h})\big)$ by only retaining the co-located secondary datum at each location $\mathbf{u}$ and assuming one of two Markov models. The algorithm *COSGSIM* allows to choose between solving a full cokriging system, or using either Markov Model 1 (MM1) or Markov Model 2 (MM2).

Algorithm 4 COSGSIM

1. If necessary, transform Z_1 and Z_2 into Gaussian variables Y_1 and Y_2, according to Eq. (3.2). Make sure that the transformed variables Y_1 and Y_2 are at least bigaussian. If they are not, another simulation algorithm should be considered, *COSISIM* for example (see 3.5).

2. Simulate variable Y_1:

   (a) Define a path visiting each node of the grid

   (b) At each node **u**, the local conditional cumulative distribution function is Gaussian, its mean is estimated by the simple cokriging estimate and its variance by the simple cokriging variance. The conditioning data consist of the original primary and secondary data and the previously simulated values

   (c) Draw a value from the previous ccdf and add it to the data set

   (d) Proceed to the next node in the path and repeat the two previous steps until all nodes are visited

3. "Back-transform" the simulated values $y_{1,1}, ..., y_{1,N}$ into $z_{1,1}, ..., z_{1,N}$:

$$z_{1,i} = F_1^{-1}\Big(G_1(y_{1,i})\Big) \quad i = 1,\ldots,N \tag{3.9}$$

   where F_1 is the target distribution function of Z_1 and G_1 is the standard normal cdf.

Example:

Seismic impedance in Stanford V is correlated to porosity with a coefficient of correlation of -0.48, see Fig. 3.10. Hence it is interesting to use seismic impedance as a secondary data to simulate porosity. Seismic information is available at every grid node, and in order to simplify the covariances modeling only the co-located seismic information is used to simulate a given node. The covariance inference problem is further simplified by assuming that the MM2 screening property (Eq. 2.10) applies. The variogram of seismic impedance is first modeled by an anisotropic exponential variogram of ranges $40/15/5$, the longest range is oriented along the X direction. The variogram of porosity is then modeled according to Eq. (2.12), with ρ_R a spherical model of ranges $40/40/5$. The complete list of input

parameters to *COSGSIM* is reproduced in Fig. 3.11. Only the second layer of Stanford V is simulated.

Fig. 3.12 shows different views of the simulated field, along with the corresponding seismic impedance and true porosity fields.

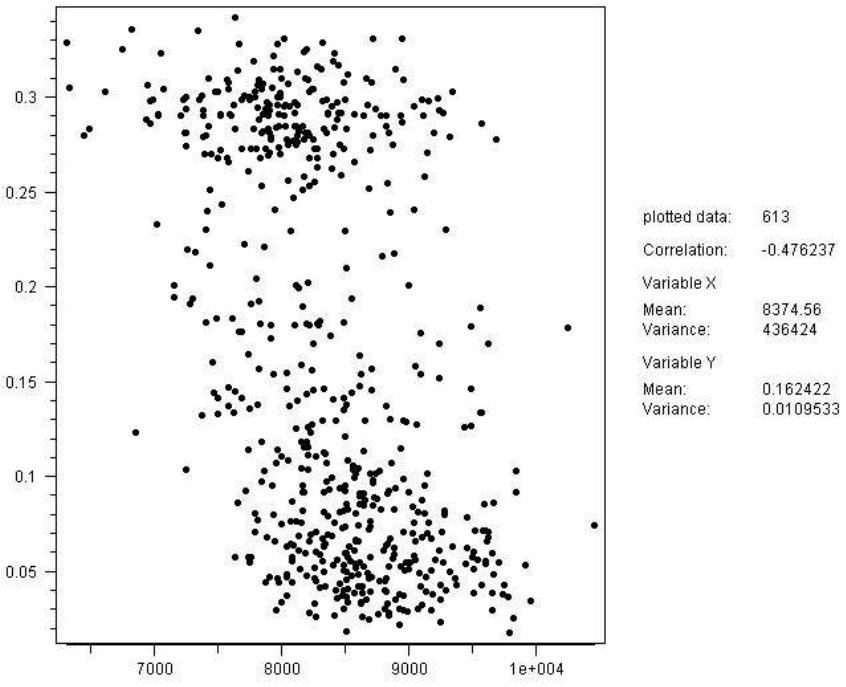
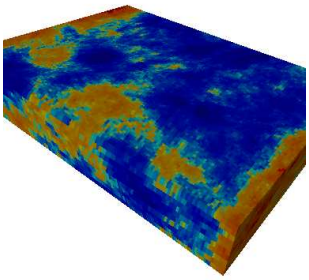


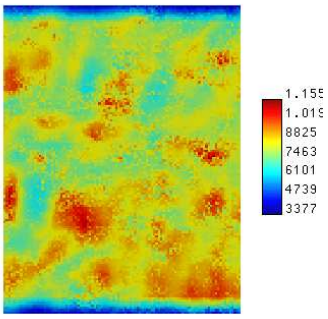
Figure 3.10: Porosity vs. seismic impedance

```
<parameters>  <algorithm name="cosgsim" />
  <Grid_Name  value="layer 2"  />
  <Property_Name  value="cosgsim_porosity" />

  <Primary_Harddata_Grid  value="layer 2 well data"  />
  <Primary_Variable  value="porosity"  />
  <Assign_Hard_Data  value="1"  />

  <Secondary_Harddata_Grid  value="layer 2"  />
  <Secondary_Variable  value="seismic impedance"  />

  <Nb_Realizations  value="5" />
  <Seed  value="14071789" />

  <Kriging_Type  value="Simple Kriging (SK)"  />
  <Cokriging_Type  value="Markov Model 2"  />

  <Max_Conditioning_Data_1  value="12" />
  <Search_Ellipsoid_1  value="70 70 6
                              0 0 0 " />

  <Variogram_C11  nugget="0" structures_count="2"  >
    <structure_1  contribution="0.25"  type="Exponential"   >
      <ranges max="40"  medium="15"  min="5"   />
      <angles x="0"  y="0"  z="0"   />
    </structure_1>
    <structure_2  contribution="0.75"  type="Spherical"   >
      <ranges max="40"  medium="40"  min="5"   />
      <angles x="0"  y="0"  z="0"   />
    </structure_2>
  </Variogram_C11>

  <MM2_Correl_Z1Z2  value="-0.5" />
  <MM2_Variogram_C22  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Exponential"   >
      <ranges max="40"  medium="15"  min="5"   />
      <angles x="0"  y="0"  z="0"   />
    </structure_1>
  </MM2_Variogram_C22>

  <Transform_Primary_Variable  value="1"  />
  <Prim_Histogram_From_Harddata  value="1"  />
  <Primary_Histogram_From_File  value="0"  />
  <Use_Min_Max  value="1"  />
  <Target_Hist_Min  value="0" />
  <Target_Hist_Max  value="0.4" />

  <Transform_Secondary_Variable  value="1"  />
  <Sec_Histogram_From_Harddata  value="1"  />
  <Secondary_Histogram_From_File  value="0"  />
</parameters>
```

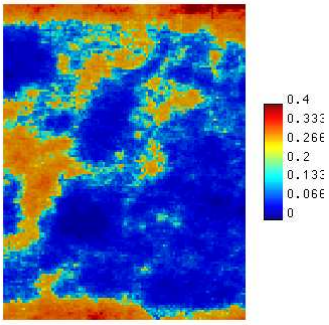
Figure 3.11: Sequential Gaussian co-simulation parameters



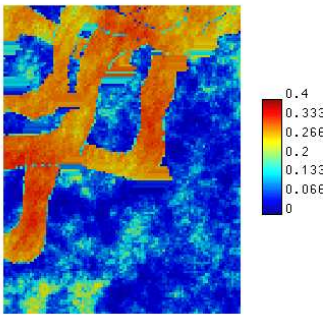
(a) 3D view



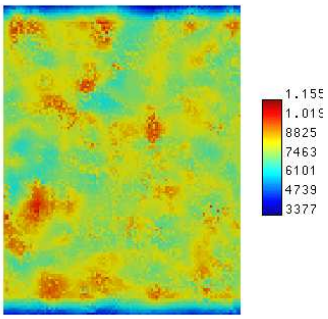
(b) Cross-section 1 - Seismic impedance



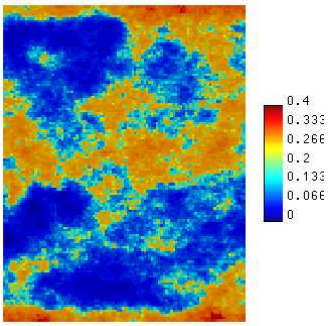
(c) Cross-section 1 - COSGSIM realization



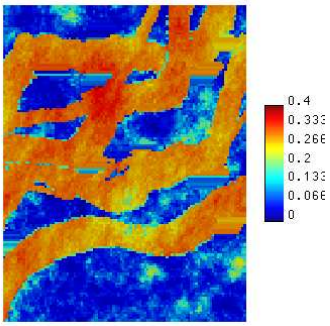
(d) Cross-section 1 - True porosity field



(e) Cross-section 2 - Seismic impedance



(f) Cross-section 2 - COSGSIM realization



(g) Cross-section 2 - True porosity field

Figure 3.12: Gaussian co-simulation of porosity, conditioned to seismic impedance

Parameters description

`Primary_Harddata_Grid:` The grid containing the primary hard data

`Primary_Variable:` The name of the property containing the primary hard data

`Assign_Hard_Data:` A flag specifying whether the hard data should be re-located on the simulation grid.

`Secondary_Harddata_Grid:` The grid containing the secondary hard data

`Secondary_Variable:` The name of the property containing the secondary hard data

`Transform_Primary_Variable:` A flag indicating whether steps 1 and 3 of **Algorithm 4** should be performed on the primary variable. Setting this flag is necessary if the primary hard data do not follow a standard normal distribution

`Prim_Histogram_From_Harddata:` If this flag is on, the distribution function of the primary variable is inferred from the primary hard data. This parameter is ignored if `Transform_Primary_Variable` is set to 0

`Primary_Histogram_From_File:` If this flag is on, the distribution of the primary variable is inferred from values read from a file. The file name is specified by `Primary_Histogram_File`. See parameter `Primary_Histogram_File` for a description of the file format and its expected content. Note that `Prim_Histogram_From_Hard` and `Primary_Histogram_From_File` are mutually exclusive

`Primary_Histogram_File:` The name of the file containing the histogram description of the primary variable. This is an ASCII file containing values drawn from the distribution of the primary variable. The values can be separated by spaces, tabulations of return carriage. The distribution (cdf) of the primary variable is computed from that list of values. This parameter is ignored if `Primary_Histogram_From_File` is set to 0

`Use_Min_Max:` A flag indicating whether we want to specify the min/max of the distribution of the primary variable. If not set, the min/max of the distribution is the min/max

of the hard data if `Target_Hist_From_Harddata` is set, or the min/max of the values in file `Target_Hist_File`

`Target_Hist_Min:` The minimum of the distribution of the primary variable. This parameter is ignored if `Use_Min_Max` is set to 0

`Target_Hist_Max:` The maximum of the distribution of the primary variable. This parameter is ignored if `Use_Min_Max` is set to 0

`Transform_Secondary_Variable:` A flag indicating whether step 1 of **Algorithm 4** should be performed on the secondary variable. Setting this flag is necessary if the secondary hard data do not follow a standard normal distribution.

`Sec_Histogram_From_Harddata:` If this flag is on, the distribution function of the secondary variable will be inferred from the secondary hard data. This parameter is ignored if `Transform_Secondary_Variable` is set to 0

`Secondary_Histogram_From_File:` If this flag is on, the distribution of the secondary variable will be inferred from values read from a file. The file name is specified by `Secondary_Histogram_File`. See parameter `Secondary_Histogram_File` for a description of the file format and its expected content. Note that `Sec_Histogram_From_Harddata` and `Secondary_Histogram_From_File` are mutually exclusive

`Secondary_Histogram_File:` The name of the file containing the histogram description. This is an ASCII file containing values drawn from the distribution. The values can be separated by spaces, tabulations of return carriage. The target distribution is computed from that list of values. This parameter is ignored if `Secondary_Histogram_From_Fi` is set to 0

`Kriging_Type:` The possible types of kriging include: Simple Kriging (SK) and Ordinary Kriging (OK)

`Cokriging_Type:` The cokriging option. Available options are: Full Cokriging, Co-located Cokriging with Markov Model 1 (MM1), and Co-located Cokriging with Markov Model 2 (MM2)

`Max_Conditioning_Data_1:` The maximum number of primary conditioning data used for each cokriging

`Search_Ellipsoid_1:` The ranges and angles defining the search ellipsoid used to find the primary conditioning data

`Variogram_C11:` The variogram of the primary variable

`Max_Conditioning_Data_2:` The maximum number of secondary conditioning data to be used for each cokriging. This parameter is only required if `Cokriging_Type` is set to Full Cokriging

`Search_Ellipsoid_2:` The ranges and angles defining the search ellipsoid used to find the secondary conditioning data. This parameter is only required if `Cokriging_Type` is set to Full Cokriging

`Variogram_C12:` The cross-variogram between the primary and secondary variables. This parameter is only required if `Cokriging_Type` is set to Full Cokriging

`Variogram_C22:` The variogram of the secondary variable. This parameter is only required if `Cokriging_Type` is set to Full Cokriging

`Correl_Z1Z2:` The coefficient of correlation between the primary and secondary variables. This parameter is only required if `Cokriging_Type` is set to MM1

`MM2_Correl_Z1Z2:` The coefficient of correlation between the primary and secondary variables. This parameter is only required if `Cokriging_Type` is set to MM2

`MM2_Variogram_C22:` The variogram of the secondary variable. This parameter is only required if `Cokriging_Type` is set to MM2. Note that the variogram of the secondary variable and the variogram of the primary variable must verify Eq. (2.12)

3.5 COSISIM

Algorithm *SISIM* simulates a variable Z conditioned only to z-values. The indicator formalism actually allows the conditioning to very diverse types of data: besides equality data,

inequality constraints or probabilistic constraints can be honored. Algorithm *COSISIM* allows to account for such types of information.

Coding information as indicator values

Consider a continuous variable $Z(\mathbf{u})$ and a discretization of its range by the K threshold values $z_1, \ldots, z_K$.

Different types of information can be coded into a vector of K indicator values $\mathbf{I} = [\mathrm{i}_1, \ldots, \mathrm{i}_\mathrm{K}]$.

Equality data

The value of Z at a given location $\mathbf{u}$ is known, equal to $z(\mathbf{u})$. The corresponding K indicator values are given by:

$$i_k = \begin{cases} 1 & \text{if } z(\mathbf{u}) \leq \mathrm{z_k} \\ 0 & \text{otherwise} \end{cases} \qquad k = 1, \ldots, K$$

Example:
If $K = 5$, and $z_1 = 1,\ z_2 = 2,\ z_3 = 3,\ z_4 = 4,\ z_5 = 5$, then $z = 3.7$ would be coded as:

$$I = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Inequality data

There are cases when the value of $z(\mathbf{u})$ is not known precisely; all that is known is an interval that contains the value, e.g $z(\mathbf{u}) \in [\mathrm{a}, \mathrm{b}]$, or $z(\mathbf{u}) \in]\mathrm{a}, +\infty[$. The indicator vector

I corresponding to $z(\mathbf{u}) \in [\mathrm{a}, \mathrm{b}[$ is given by:

$$i_k = \begin{cases} 0 & \text{if } z_k < a \\ \text{not defined} & \text{if } z_k \in [a, b[\qquad k = 1, \ldots, K \\ 1 & \text{if } z_k \geq b \end{cases}$$

Example:
If $K = 5$, and $z_1 = 1,\ z_2 = 2,\ z_3 = 3,\ z_4 = 4,\ z_5 = 5$, then $z \in]2.1, 4.9]$ would be coded as:

$$I = \begin{bmatrix} 0 \\ 0 \\ ? \\ ? \\ 1 \end{bmatrix}$$

where ? denotes a value that is not defined (missing data value)

Soft data

Information that are only indirectly related to variable Z can also be coded into "indicator" values, or more precisely, into prior probability values.

Consider a variable $Y(\mathbf{u})$ correlated with Z. Y could be the permeability of a rock and Z its porosity. If the conditional prior probability function $F(Z(\mathbf{u}) \leq \mathrm{z_k} \mid \mathrm{Y}(\mathbf{u}))$ is known, it can be used as an indicator coding of the influence of Y on Z:

$$I_k(\mathbf{u}) = \mathrm{F}\Big(\mathrm{Z}(\mathbf{u}) \leq \mathrm{z_k} \mid \mathrm{Y}(\mathbf{u})\Big) \qquad \mathrm{k} = 1, \ldots, \mathrm{K}$$

$I_k(\mathbf{u})$ is not anymore equal to 0 or 1 only but is a probability value, valued in $[0, 1]$, reflecting the uncertainty on $Z(\mathbf{u})$ given the information $Y(\mathbf{u})$.

The prior probability function F could be estimated in different ways. One possibility might be to use the experimental conditional distribution: If Y is a continuous variable and $y_1, \ldots, y_{K'}$ a discretization of its range, then

$$F^*\Big(Z \leq z_k \mid Y \in [y_l, y_{l+1}]\Big) = \frac{1}{\sum_{\mathbf{u}_\alpha \in \mathcal{U}} i(\mathbf{u}_\alpha, [\mathrm{y_l}, \mathrm{y_{l+1}}])} \sum_{\mathbf{u}_\alpha \in \mathcal{U}} i(\mathbf{u}_\alpha, \mathrm{z_k})\ \mathrm{i}(\mathbf{u}_\alpha, [\mathrm{y_l}, \mathrm{y_{l+1}}])$$

$\mathcal{U}$ is the set of locations where both Z and Y are known and:

$$i(\mathbf{u}_\alpha, \mathrm{z_k}) = \begin{cases} 1 & \text{if } Z(\mathbf{u}_\alpha) \leq \mathrm{z_k} \\ 0 & \text{otherwise} \end{cases}$$

$$i(\mathbf{u}_\alpha, [\mathrm{y_l}, \mathrm{y_{l+1}}]) = \begin{cases} 1 & \text{if } Y(\mathbf{u}_\alpha) \in [\mathrm{y_l}, \mathrm{y_{l+1}}] \\ 0 & \text{otherwise} \end{cases}$$

If variable Y is categorical, taking the categorical values $\{1, \ldots, K'\}$, then the prior distribution function is given by:

$$F^*\Big(Z \leq z_k \mid Y = k'\Big) = \frac{1}{\sum_{\mathbf{u}_\alpha \in \mathcal{U}} i(\mathbf{u}_\alpha, \mathrm{k'})} \sum_{\mathbf{u}_\alpha \in \mathcal{U}} i(\mathbf{u}_\alpha, \mathrm{z_k})\, \mathrm{i}(\mathbf{u}_\alpha, \mathrm{k'}) \tag{3.10}$$

with $i(\mathbf{u}_\alpha, \mathrm{k'})$ defined as:

$$i(\mathbf{u}_\alpha, \mathrm{k'}) = \begin{cases} 1 & \text{if } Y(\mathbf{u}_\alpha) = \mathrm{k'} \\ 0 & \text{otherwise} \end{cases}$$

Accounting for soft data: the Markov-Bayes algorithm

Indicator coding provides a flexible means of accounting for different types of information. The "indicator" coded soft data can be interpreted as a secondary variable Y and accounted for by solving a cokriging system. The (ordinary) cokriging estimator is given by:

$$I^*(\mathbf{u}, \mathrm{z_k}) = \sum_\alpha \lambda_\alpha \mathrm{I}(\mathbf{u}_\alpha), \mathrm{z_k}) + \sum_\beta \lambda_\beta \mathrm{Y}\Big(\mathbf{u}_\beta, \mathrm{z_k}\Big)$$

with $Y\Big(\mathbf{u}_\beta, \mathrm{z_k}\Big) = \mathrm{F}^*\Big(\mathrm{Z}(\mathbf{u}) \leq \mathrm{z_k} \mid \mathrm{Y}\Big)$. In all rigor, the cokriging estimator should include indicator and soft data at thresholds other than z_k. However Goovaerts (1994) showed that in practice those terms could be neglected.

Solving this cokriging system calls for the knowledge of two covariance functions and one cross-covariance function for each threshold z_k. The inference and modeling work can be alleviated if the following Markov property is assumed:

$$Prob\Big(Z(\mathbf{u}') \leq \mathrm{z_k} \mid \mathrm{i}(\mathbf{u}, \mathrm{z_k}), \mathrm{y}(\mathbf{u}, \mathrm{z_k})\Big) = \mathrm{Prob}\Big(\mathrm{Z}(\mathbf{u}') \leq \mathrm{z_k} \mid \mathrm{i}(\mathbf{u}, \mathrm{z_k})\Big) \qquad \forall \mathbf{u}, \mathbf{u}, \mathrm{z_k} \tag{3.11}$$

With this screening ("Markov") hypothesis, it can be shown [Zhu and Journel, 1993] that:

$$C_Y(\mathbf{h}, \mathrm{z_k}) = \begin{cases} |B(z_k)|\, C_I(\mathbf{h}, \mathrm{z_k}) & \text{if } \|\mathbf{h}\| = 0 \\ B(z_k)^2\, C_I(\mathbf{h}, \mathrm{z_k}) & \text{if } \|\mathbf{h}\| > 0 \end{cases} \tag{3.12}$$

$$C_{IY}(\mathbf{h}, \mathrm{z_k}) = B(z_k)\, C_I(\mathbf{h}, \mathrm{z_k}) \tag{3.13}$$

where $B(z_k)$ is the difference between two conditional expectations:

$$B(z_k) = m_1(z_k) - m_0(z_k) \quad \in [-1, 1]$$

defined as:

$$m_1(z_k) = E\Big(Y(\mathbf{u}, \mathrm{z_k}) | \mathrm{I}(\mathbf{u}, \mathrm{z_k}) = 1\Big)$$

$$m_0(z_k) = E\Big(Y(\mathbf{u}, \mathrm{z_k}) | \mathrm{I}(\mathbf{u}, \mathrm{z_k}) = 0\Big)$$

Updating of the prior probabilities $Y(\mathbf{u}, \mathrm{z_k})$ under that Markov hypothesis is referred to as the "Markov-Bayes algorithm".

Example:

Section 3.3 showed an example of indicator simulation of sand and shale facies in the layer 2 of Stanford V. In that example, the simulation was only constrained to facies well data, and the sand bodies, while having the long continuity of the true channels, failed to reproduce their sinuosity. The result of the indicator simulation can be improved by integrating additional information, such as seismic impedance.

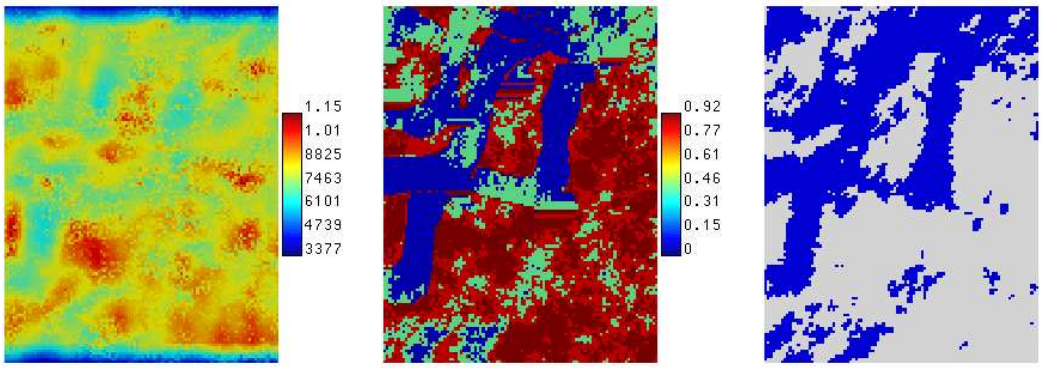
As mentioned in the discussion on information coding into indicator values, seismic impedance has to be transformed into a prior probability $P(I(\mathbf{u}) = 0 \mid \mathrm{Y}(\mathbf{u}))$ of observing the shale facies given a seismic impedance datum. The prior probabilities are computed according to Eq. (3.10). Fig. 3.14 shows the obtained probability maps for two cross-sections.

In order to simplify the variogram inference, the Markov property (3.11) is assumed. Parameters $B(0)$ and $B(1)$ (Eq. (3.12)), inferred from the well data, are both equal to 0.5. The complete parameter file for the run is reproduced in Fig. 3.13.

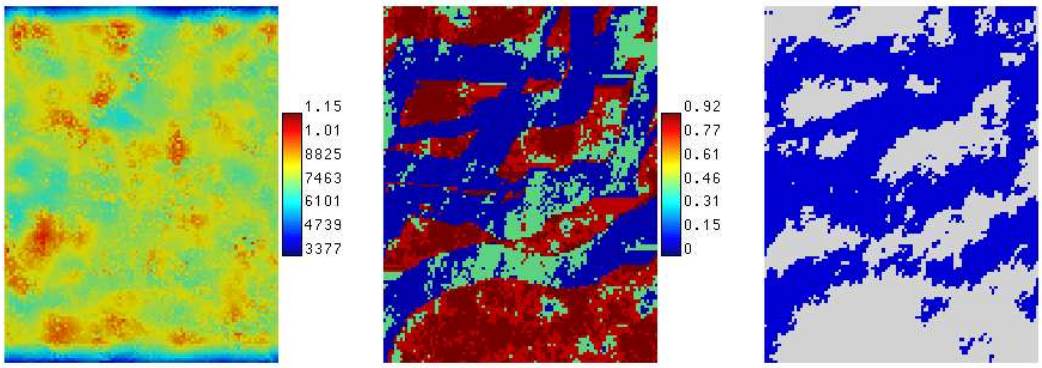
Fig. 3.14 shows two cross-sections of a sample realization. Due to the extremely high quality of the seismic information (the prior probability maps clearly identify the channels), the channels are very well reproduced.

```
<parameters>  <algorithm name="cosisim" />
  <Grid_Name  value="layer 2"  />
  <Property_Name  value="cosisim_facies" />

  <Nb_Realizations  value="5" />
  <Seed  value="14071789" />

  <Categorical_Variable_Flag  value="1"  />
  <Nb_Indicators  value="2" />
  <Marginal_Probabilities  value="0.57 0.43" />

  <Kriging_Type  value="Simple Kriging"  />
  <Median_Ik_Flag  value="1"  />

  <Primary_Harddata_Grid  value="well data"  />
  <Primary_Indicators count="2"  value="shale indicator; sand indicator"  />
  <Max_Conditioning_Data_Primary  value="12" />
  <Search_Ellipsoid_Primary  value="60 60 7
                                    0 0 0 " />

  <Secondary_Harddata_Grid  value="layer 2"  />
  <Secondary_Indicators count="2"   value="P(I=0|Y); P(I=1|Y)"  />
  <Max_Conditioning_Data_Secondary  value="12" />
  <Search_Ellipsoid_Secondary  value="20 20 4
                                      0 0 0 " />

  <Bz_Values  value="0.5  0.5" />
  <Variogram_Median_Ik  nugget="0" structures_count="1"  >
    <structure_1  contribution="1"  type="Spherical"   >
      <ranges max="40"  medium="25"  min="5"   />
      <angles x="45"  y="0"  z="0"   />
    </structure_1>
  </Variogram_Median_Ik>
</parameters>
```

Figure 3.13: Sequential indicator co-simulation parameters



(a) Seismic impedance (b) $P(I = 0 \mid Y)$ (c) COSISIM realization



(d) Seismic impedance (e) $P(I = 0 \mid Y)$ (f) COSISIM realization

Figure 3.14: Indicator co-simulation of facies, conditioned to seismic impedance. Each line of plots corresponds to a different cross-section. Each line shows the seismic impedance, the corresponding prior probability $P(I = 0 \mid Y)$ ($I = 0$ corresponds to the shale facies) and the COSISIM realization. The shale facies is in white, sand in blue.

Parameters description

`Primary_Harddata_Grid:` The grid containing the primary hard data

`Primary_Indicators:` The name of the properties containing the indicator values for the primary variable. Exactly K properties must be specified.

`Secondary_Harddata_Grid:` The grid containing the soft data. This parameter is optional

`Secondary_Indicators:` The name of the properties containing the indicator values for the soft data. Exactly K properties must be specified. This parameter is optional

`Categorical_Variable_Flag:` A flag indicating whether the variable to be simulated is categorical or continuous. If the flag is on (equal to 1), the variable is categorical

`Nb_Indicators:` The number K of indicator variables or the number K of categories

`Thresholds:` The threshold values $z_1, \ldots, z_K$ defining each indicator variable. This is only needed for continuous variables. The threshold values are separated by spaces. This parameter is ignored if `Categorical_Variable_Flag` is set to 1

`Marginal_Probabilities:` The marginal probabilities of each indicator. The probability values are separated by one or more space

`Kriging_Type:` The possible types of kriging are: Simple Kriging (SK) and Ordinary Kriging (OK)

`Median_Ik_Flag:` A flag indicating whether median IK should be performed

`Full_Ik_Flag:` A flag indicating whether full IK should be performed. Note that `Median_Ik_Flag` and `Full_Ik_Flag` are mutually exclusive

`Max_Conditioning_Data_Primary:` The maximum number of primary conditioning data to be used for each cokriging

`Search_Ellipsoid_Primary:` The ranges and angles defining the search ellipsoid used to find the primary conditioning data

`Max_Conditioning_Data_Secondary:` The maximum number of soft conditioning data to be used for each cokriging

`Search_Ellipsoid_Secondary:` The ranges and angles defining the search ellipsoid used to find the soft conditioning data

`Bz_Values:` The B values for each indicator variogram (see the Markov-Bayes algorithm). The values are separated by spaces. This parameter is not required if no soft data are used

`Variogram_Median_Ik:` The single indicator variogram model used for median IK. If `Median_Ik_Flag` is off this parameter is ignored

`Variogram_Full_Ik:` The indicator variogram models (one for each indicator) of each indicator variable. If `Median_Ik_Flag` is on this parameter is ignored

3.6 SNESIM

Algorithms based on two-points statistics, e.g. all algorithms based on kriging, fail to model complex curvilinear heterogeneities. It is possible to produce images with very different structures (e.g. some with channels stretching from one end of the image to the other, and some featuring non-overlapping ellipsis of "small" dimensions) which all share the same two-point statistics (i.e. covariance) [Strebelle, 2000]: two point-statistics are not enough to fully characterize complex (low-entropy) structures, and one must rely on higher order statistics to model those structures.

Algorithm *SNESIM* implements a sequential simulation algorithm, called single normal equation simulation [Strebelle, 2000], that simulates categorical variables using multiple-point statistics. In *SNESIM*, the multiple-point statistics are not provided by an analytical function but are directly borrowed from a training image. The training image depicts the type of structures the simulated realizations should exhibit, and it needs not be constrained

to any local data such as sampled values or well data. Training images can conveniently be generated by unconditional object-based simulations [Tjelmeland, 1996; Holden et al., 1998; Chilès and Delfiner, 1999].

Algorithm 5 describes the simplest version of the *snesim* algorithm.

Step 2(b) of **Algorithm 5** is critical: it requires the search of all locations $\mathbf{u}_\alpha$ of the training image where $D_j(\mathbf{u}_\alpha) = \mathrm{D_j}(\mathbf{u})$, $1 \leq j \leq J$. Performing an exhaustive search for each location to be simulated is not practical. Instead *SNESIM* uses a tree data structure (called "search-tree") to efficiently perform the search. The search-tree stores all the different data events $D_j(\mathbf{u}_\alpha)$, $j = 1, \ldots, J$, found at all locations $\mathbf{u}_\alpha$ of the training image, and allows to retrieve the conditional probability distribution of step 2(c) in $O(log(J))$. This speed comes at the cost of a possibly large RAM memory demand.

Let N_{TI} be the total number of locations in the training image. Since for a given data event size j, there can not be more than N_{TI} different data events in the training image, a (crude) upper-bound of the memory demand of the search tree is:

$$\text{Memory Demand} \leq \sum_{j=1}^{J} min(K^j, N_{TI})$$

Multiple grid simulation

Reproducing large scale structures would call for using a large template. However because of memory constraints, it is not practical to work with too large templates. A work-around is provided by the multiple-grid approach [Tran, 1994].

Consider a 3-D Cartesian grid $\mathcal{G}$. Define $\mathcal{G}^{(l)}$ as the sub-set of $\mathcal{G}$ such that: $\mathcal{G}^{(0)} = \mathcal{G}$ and $\mathcal{G}^{(l)}$ is obtained by down-sampling $\mathcal{G}^{(l-1)}$ by a factor of 2 in all 3 coordinate directions: $\mathcal{G}^{(l)}$ is the sub-set of $\mathcal{G}^{(l-1)}$ obtained by retaining every other node of $\mathcal{G}^{(l-1)}$. $\mathcal{G}^{(l)}$ is often called the l^{th} level grid.

Algorithm 6 describes how multiple grids are used in program *SNESIM*

Marginal distribution reproduction

It is sometimes desirable that the histogram of the simulated variable be close to a given target distribution, e.g. the sample histogram. There is however no constraint in *SNESIM*

Algorithm 5 Simple Snesim

Consider a categorical variable $Z(\mathbf{u})$ valued in $\{1, \ldots, K\}$.

1. Define a geometrical template (or "window") $T_J = \{\mathbf{h_1}, \ldots, \mathbf{h_J}\}$, $\mathbf{h}_\mathrm{j}$ is a vector in $\mathbb{R}^3$, and a path visiting all locations to be simulated

2. For each location $\mathbf{u}$ along the path:

   (a) Retrieve the conditioning data event $D_J(\mathbf{u})$ defined by template T_J:

   $$D_J(\mathbf{u}) = \{\mathrm{z}(\mathbf{u} + \mathbf{h_1}), \ldots, \mathrm{z}(\mathbf{u} + \mathbf{h_J})\}$$

   (b) Find all locations $\mathbf{u}_\alpha$, $\alpha = 1, \ldots, n$, of the training image such that $D_J(\mathbf{u}_\alpha) = \mathrm{D_J}(\mathbf{u})$, i.e. such that:

   $$z(\mathbf{u} + \mathbf{h_j}) = \mathrm{z}(\mathbf{u}_\alpha + \mathbf{h_j}) \quad \forall \mathrm{j} \in \{1, \ldots, \mathrm{J}\}$$

   If the number n of such locations is lesser than a fixed threshold n_{min}, retrieve the smaller data event D_{J-1} defined by T_{J-1} and repeat the search. This step is repeated until $n \geq n_{min}$. Call J' the largest template size for which $n \geq n_{min}$.

   (c) The conditional probability distribution $P(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})) = \mathrm{E}(\mathrm{I}(\mathbf{u}, \mathrm{k}) \mid \mathrm{D_J}(\mathbf{u}))$, $k = 1, \ldots, K$ is modeled by the experimental conditional expectation $E^*(I(\mathbf{u}, \mathrm{k}) \mid \mathrm{D_{J'}}(\mathbf{u}))$:

   $$\begin{aligned} P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big) &\simeq P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_{J'}}(\mathbf{u})\big) \\ &\simeq E^*(I(\mathbf{u}, \mathrm{k}) \mid \mathrm{D_{J'}}(\mathbf{u})) \\ &= \frac{1}{n} \sum_{\alpha=1}^{n} i\big(z(\mathbf{u}_\alpha), \mathrm{k}\big) \end{aligned}$$

   where $i\big(z(\mathbf{u}_\alpha), \mathrm{k}\big)$ is the indicator function:

   $$i\big(z(\mathbf{u}_\alpha), \mathrm{k}\big) = \begin{cases} 1 & \text{if } z(\mathbf{u}_\alpha) = \mathrm{k} \\ 0 & \text{otherwise} \end{cases}$$

   Draw a simulated value from that conditional distribution and add it to the data set

3. Repeat step 2 until all locations in the path are visited

Algorithm 6 SNESIM with multiple grids

1. Choose the number L of multiple grids to consider.
2. Start with the coarsest grid $\mathcal{G}^{(L)}$.
3. Build a new geometrical template $T_J^{(L)}$ by re-scaling template T_J:

$$T_J^{(L)} = \{2^{L-1}\mathbf{h}_1, \ldots, 2^{\mathrm{L}-1}\mathbf{h}_{\mathrm{J}}\}$$

   Template $T_J^{(L)}$ has the same number of vectors as T_J but has a much greater extent than T_J, hence will be able to capture large-scale variations more efficiently without increasing the search tree size.
4. Build the search tree using the training image and template $T_J^{(L)}$
5. Simulate all nodes of $\mathcal{G}^{(L)}$ as in **Algorithm 5**
6. All the nodes of $\mathcal{G}^{(L)}$ are now part of the data set. Repeat steps 3 to 5 on the next finer grid $\mathcal{G}^{(L-1)}$
7. Loop until all nested multiple grids have been simulated

as described in **Algorithm 5** or **Algorithm 6** to ensure that any target distribution will be reproduced. *SNESIM* uses an updating of the conditional distribution function computed at each location (see step 2(c) of **Algorithm 5**) to control the histogram of the simulated values.

Let p_k^c, $k = 1, \ldots, K$, denote the proportions of values in class k simulated so far, and p_k^t, $k = 1, \ldots, K$, be the target proportions. Step 2(c) of **Algorithm 5** is modified as follows:

1. Compute the conditional probability distribution as originally described in step 2(c) of **Algorithm 5**.
2. Update the computed probabilities $P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big)$ into probabilities $P^*\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big)$ with:

$$P^*\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big) = \mathrm{P}\big(\mathrm{Z}(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big) + \frac{\omega}{1-\omega} * (\mathrm{p_k^t} - \mathrm{p_k^c})$$

where $\omega \in [0,1[$ is a control parameter. If $\omega = 0$, no updating is performed. Conversely, if $\omega \to 1$, reproducing the target distribution entirely controls the simulation process.

If $P^*\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big) \notin [0,1]$ it is reset to the closest bound. All updated probability values are then rescaled to sum up to 1:

$$P^{**}\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big) = \frac{1}{\sum_{\mathrm{k}=1}^{\mathrm{K}} \mathrm{P}^*\big(\mathrm{Z}(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big)} \mathrm{P}^*\big(\mathrm{Z}(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big)$$

Conditioning to soft data

SNESIM can also account for a secondary variable $Y(\mathbf{u})$, for example a seismic amplitude pattern centered on **u**. The first step is to turn $Y(\mathbf{u})$ into a vector of conditional probabilities (see the paragraph on indicator coding of soft data in section 3.5):

$$\begin{bmatrix} P\big(Z(\mathbf{u}) = 1 \mid \mathrm{Y}(\mathbf{u})\big) \\ \vdots \\ P\big(Z(\mathbf{u}) = \mathrm{K} \mid \mathrm{Y}(\mathbf{u})\big) \end{bmatrix}$$

The conditional distribution function of step 2(c) of **Algorithm 5**, $P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big)$, is then updated into $P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u}), \mathrm{Y}(\mathbf{u})\big)$ using the formula proposed by Journel (2002). Define the four quantities:

$$\begin{aligned} x &= \frac{1 - P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u}), \mathrm{Y}(\mathbf{u})\big)}{P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u}), \mathrm{Y}(\mathbf{u})\big)} \\ a &= \frac{1 - P\big(Z(\mathbf{u}) = \mathrm{k}\big)}{P\big(Z(\mathbf{u}) = \mathrm{k}\big)} \\ b &= \frac{1 - P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big)}{P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big)} \\ c &= \frac{1 - P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{Y}(\mathbf{u})\big)}{P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{Y}(\mathbf{u})\big)} \end{aligned}$$

Under the hypothesis of conditional independence:

$$P\big(D_J(\mathbf{u}), \mathrm{Y}(\mathbf{u}) \mid \mathrm{Z}(\mathbf{u})\big) = \mathrm{P}\big(\mathrm{D_J}(\mathbf{u}) \mid \mathrm{Z}(\mathbf{u})\big)\, \mathrm{P}\big(\mathrm{Y}(\mathbf{u}) \mid \mathrm{Z}(\mathbf{u})\big) \tag{3.14}$$

the updated probabilities are given by:

$$x = \frac{b.c}{a} \tag{3.15}$$

Step 2(c) of **Algorithm 5** is then modified as follows:

1. Estimate the probabilities $P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big)$ as described in **Algorithm 5**
2. Compute the updated probabilities $P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u}), \mathrm{Y(u)}\big)$ using Eq. (3.15)
3. Draw a realization from the updated distribution function

Accounting for local non-stationarity

The training image should be reasonably stationary so that meaningful statistics can be inferred from it. It is however possible to introduce some non-stationarity in the simulation by accounting for local rotation information. Call $R(\mathbf{u})$ the rotation angle about the (vertical) Z-axis at location **u**. In *SNESIM* the rotation axis is always the Z-axis. The angle $R(\mathbf{u})$ is restricted to take only a finite number of different values $r_1, \ldots, r_M$.

Step 2(a) of **Algorithm 5** could be modified as follows to account for R:

1. Define a new geometrical template $RT_J(\mathbf{u})$ by rotating T_J by angle $r(\mathbf{u})$:

$$RT_J = \{\Theta(\mathbf{u})\mathbf{h}_1, \ldots, \Theta(\mathbf{u})\mathbf{h}_1\}$$

   where $\Theta(\mathbf{u})$ is the rotation matrix:

$$\Theta(\mathbf{u}) = \begin{bmatrix} \cos(r(\mathbf{u})) & \sin(r(\mathbf{u})) & 0 \\ -\sin(r(\mathbf{u})) & \cos(r(\mathbf{u})) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. Retrieve the conditioning data event $D_J(\mathbf{u})$ defined by template $RT_J(\mathbf{u})$

This solution, while elegant, yields disappointing results because of aliasing issues caused by the rotation of template T_J.

Those aliasing issues are alleviated if instead of rotating template T_J the entire training image is rotated and scanned with T_J.

Algorithm 7 Snesim with locally varying azimuth

Consider a categorical variable $Z(\mathbf{u})$ valued in $\{1, \ldots, K\}$.

1. Define a geometrical template $T_J = \{\mathbf{h_1}, \ldots, \mathbf{h_J}\}$ and a path visiting all the locations to be simulated

2. Compute the rotated training images RTI_m, m=1,...,M corresponding to each angle $r_1, \ldots, r_M$. Build the corresponding M search trees, using template T_J

3. For each location **u** in the path:

   (a) Retrieve the conditioning data event $D_J(\mathbf{u})$ defined by template T_J:

   $$D_J(\mathbf{u}) = \{\mathrm{z}(\mathbf{u} + \mathbf{h_1}), \ldots, \mathrm{z}(\mathbf{u} + \mathbf{h_J})\}$$

   (b) Find all locations $\mathbf{u}_\alpha$, $\alpha = 1, \ldots, n$, of the training image rotated by angle $r(\mathbf{u})$, $RTI_{r(\mathbf{u})}$, such that $D_J(\mathbf{u}_\alpha) = \mathrm{D_J}(\mathbf{u})$, i.e. such that:

   $$z(\mathbf{u} + \mathbf{h_j}) = \mathrm{z}(\mathbf{u}_\alpha + \mathbf{h_j}) \quad \forall \mathrm{j} \in \{1, \ldots, \mathrm{J}\},\ \mathbf{u}_\alpha \in \mathrm{RTI_{r(u)}}$$

   If the number n of such locations is lesser than a fixed threshold n_{min}, retrieve the smaller data event D_{J-1} defined by T_{J-1} and repeat the search. This step is repeated until $n \geq n_{min}$. Call J' the largest template size for which $n \geq n_{min}$.

   (c) The conditional probability distribution $P(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u}))$, $k = 1, \ldots, K$ is modeled as follows:

   $$\begin{aligned} P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_J}(\mathbf{u})\big) &= P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{D_{J'}}(\mathbf{u})\big) \\ &= \frac{1}{n} \sum_{\alpha=1}^{n} i\big(z(\mathbf{u}_\alpha), \mathrm{k}\big) \end{aligned}$$

   where $i\big(z(\mathbf{u}_\alpha), \mathrm{k}\big)$ is the indicator function:

   $$i\big(z(\mathbf{u}_\alpha), \mathrm{k}\big) = \begin{cases} 1 & \text{if } z(\mathbf{u}_\alpha) = \mathrm{k} \\ 0 & \text{otherwise} \end{cases}$$

   Draw a realization from that conditional distribution, which is then added to the data set

4. Repeat step 2 until all locations in the path have been visited

The new snesim algorithm is described in **Algorithm 7**

This solution which is the one implemented in algorithm *SNESIM* can be very memory demanding as one search tree has to be built for each rotation angle $r_1, \ldots, r_M$. However, practice has shown that it is possible to generate fairly complex models while using a very limited number of angles: about 5 angle values should be sufficient in most cases

Example:
The sand/shale facies simulation of section 3.3 is repeated using *SNESIM*. Cross-sections of the 3-D, 100x130x8 training image are shown on Fig. 3.16. Four multiple grids are used, and the template contains 100 nodes. The complete set of parameters is reproduced in Fig. 3.15. Fig. 3.17 shows different views of a sample realization. Cross-sections (e) and (h) of Fig. 3.17 correspond to the bottom of the channels of cross-sections (f) and (i), which explains the noticeable lack on continuity of the channels in those sections. The channel proportion in the realization of Fig. 3.17 is 35%, compared to the 30% target proportion.

Notice the sharp difference with the *SISIM* realization (Fig. 3.9): the sand bodies are crisp are reproduce the continuity and sinuosity of the reference channels (Fig. 1.15).

Influence of the template size

One of the key *SNESIM* parameter is the size of the template: the bigger the template, the more structure it can capture. Fig. 3.18 shows the results obtained by increasing the number of template nodes. In all cases, no multiple grids are used, and the other parameters are those reproduced on Fig. 3.15.

Even with 200 nodes in the template, the channel structures are very poorly reproduced if no multiple grids are used. 200 nodes indeed constitute a template of very limited extent, e.g. 7x7x4 grid cells, making it impossible to capture structures that are continuous across 100 grid cells.

Influence of the number of multiple grids

The template size is limited by the amount of RAM available, and it is necessary to use the multiple-grid simulation technique to keep the template size low, yet capture large scale structures. Fig. 3.19 shows the influence of the number of multiple-grids on the quality of

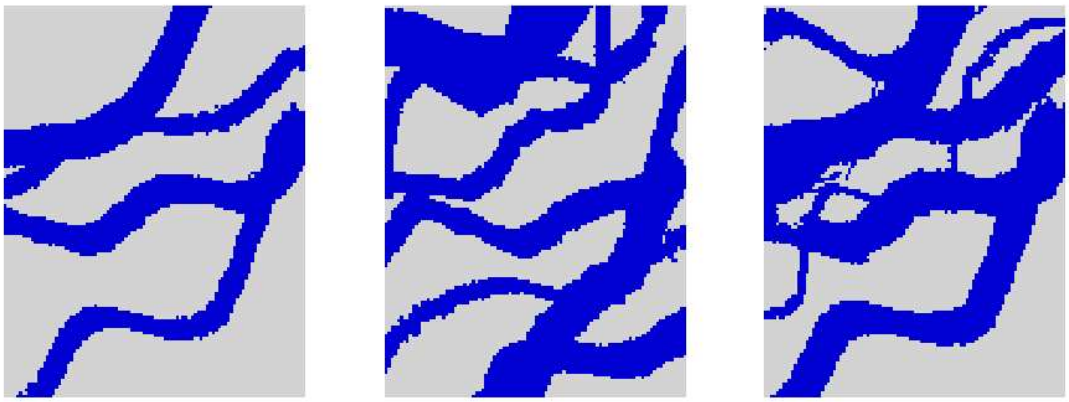
```
<parameters>  <algorithm name="snesim" />
  <GridSelector_Sim  value="layer 2"  />
  <Property_Name_Sim  value="snesim_facies" />

  <Nb_Realizations  value="5" />
  <Seed  value="211175" />

  <PropertySelector_Training  grid="TI"   property="facies (2)"  />
  <Nb_Facies  value="2" />
  <Marginal_Cdf  value="0.7  0.3" />

  <Max_Cond  value="100" />
  <Search_Ellipsoid  value="10 10 3
                            0 0 0 " />

  <Hard_Data  grid="well data"   property="facies (2)"  />

  <Use_ProbField  value="0"  />
  <Use_Rotation  value="0"  />
  <Use_Affinity  value="0"  />

  <Cmin  value="1" />
  <Constraint_Marginal_ADVANCED  value="0.5" />
  <Nb_Multigrids_ADVANCED  value="4" />
  <Subgrid_choice  value="1"  />
  <Previously_simulated  value="4" />
</parameters>
```

Figure 3.15: SNESIM parameters

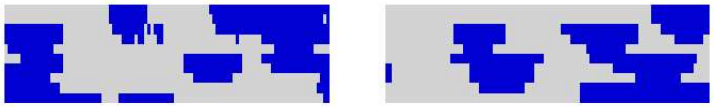
the realizations. The parameter used are those shown on Fig. 3.15, except for the number of multiple grids and the template size, which was set to 30.

The use of multiple grids drastically improves the reproduction of large scale structures, even with small, 30-nodes template.

The same sensitivity analysis is repeated with a 100-nodes template in Fig. 3.20, and a 200-nodes template in Fig. 3.21. In this case study, increasing the template size has little influence on the quality of the reproduction of large scale structures.

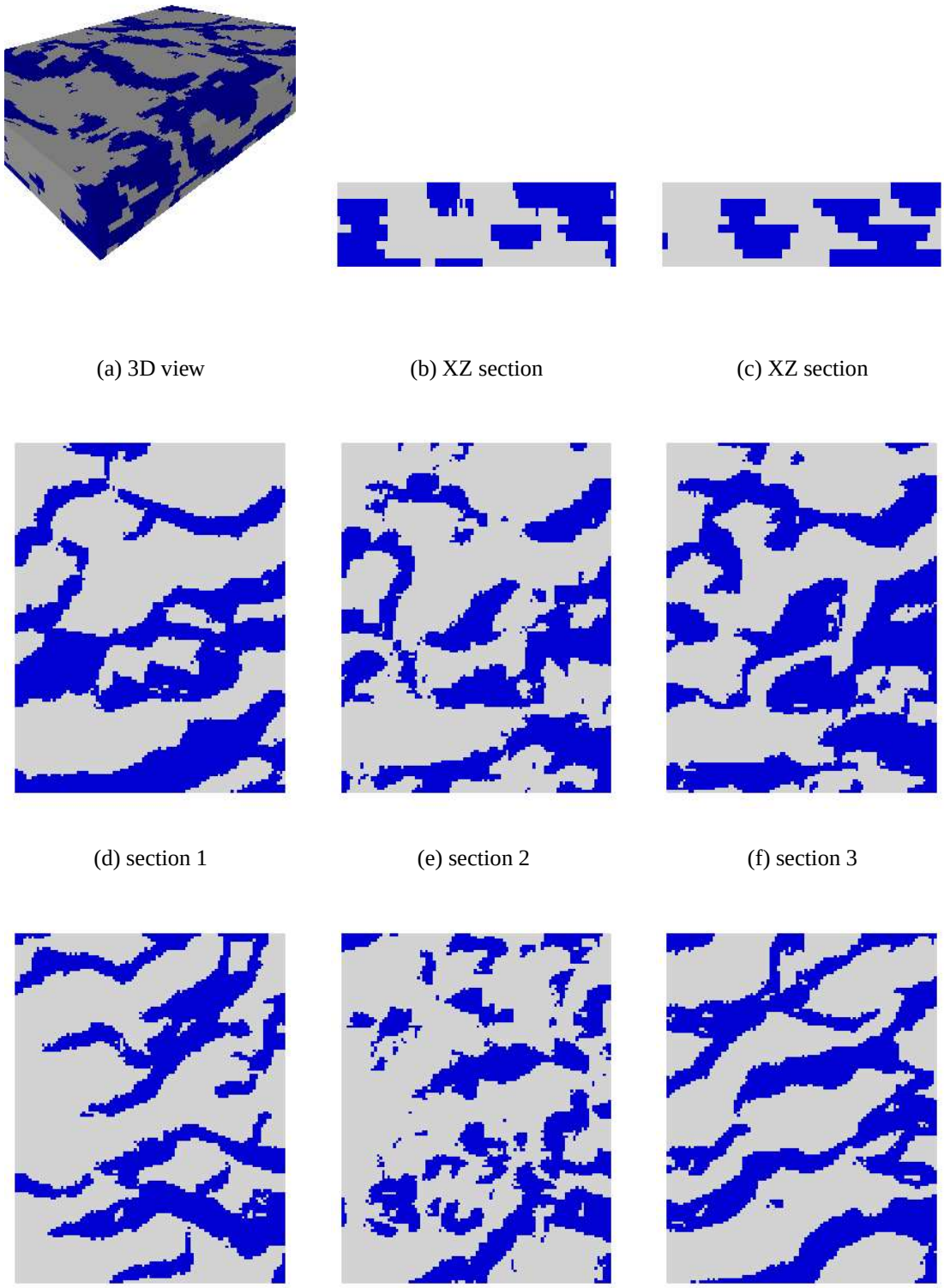


(a) XY cross-section (b) XY cross-section (c) XY cross-section



(d) XZ cross-section (e) XZ cross-section

Figure 3.16: Cross-sections of the training image

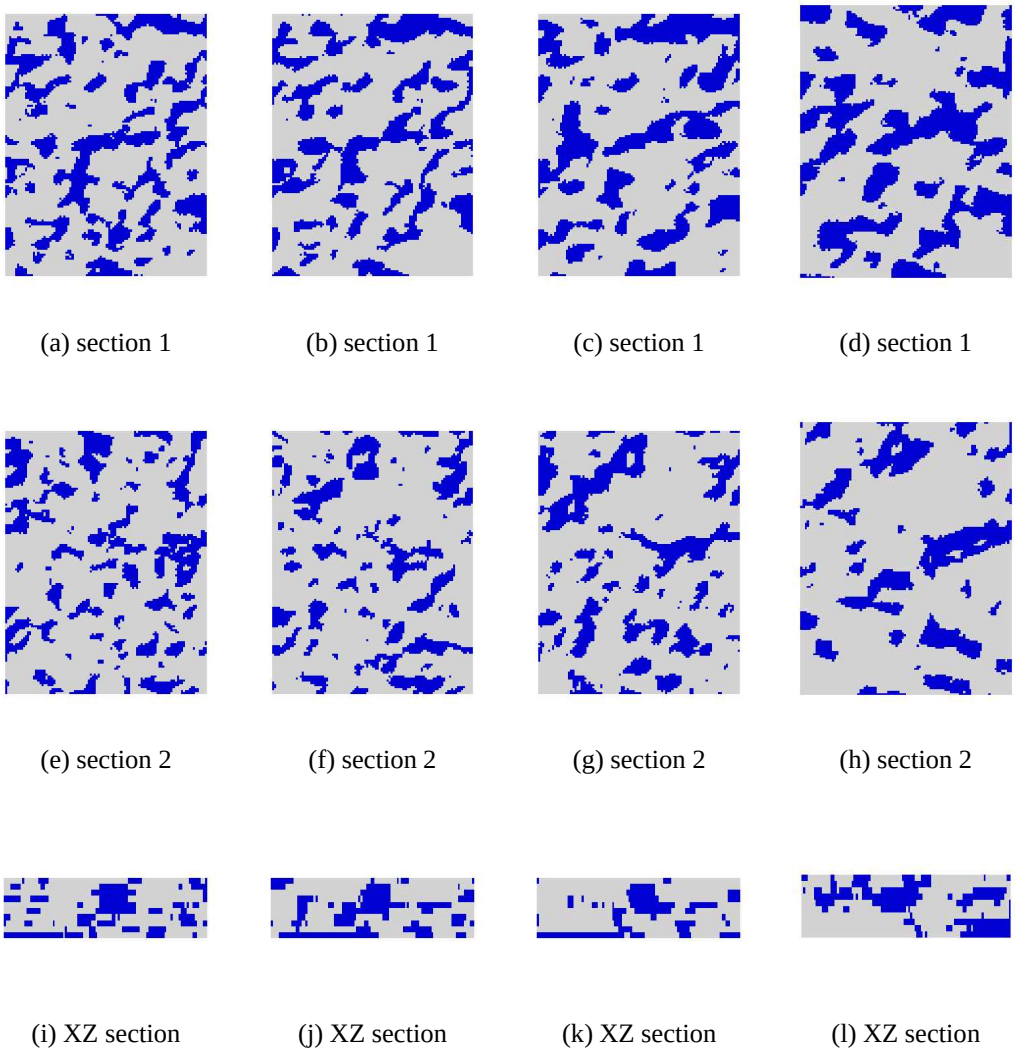


(a) 3D view (b) XZ section (c) XZ section

(d) section 1 (e) section 2 (f) section 3

(g) section 5 (h) section 8 (i) section 9

Figure 3.17: One sample *SNESIM* realization

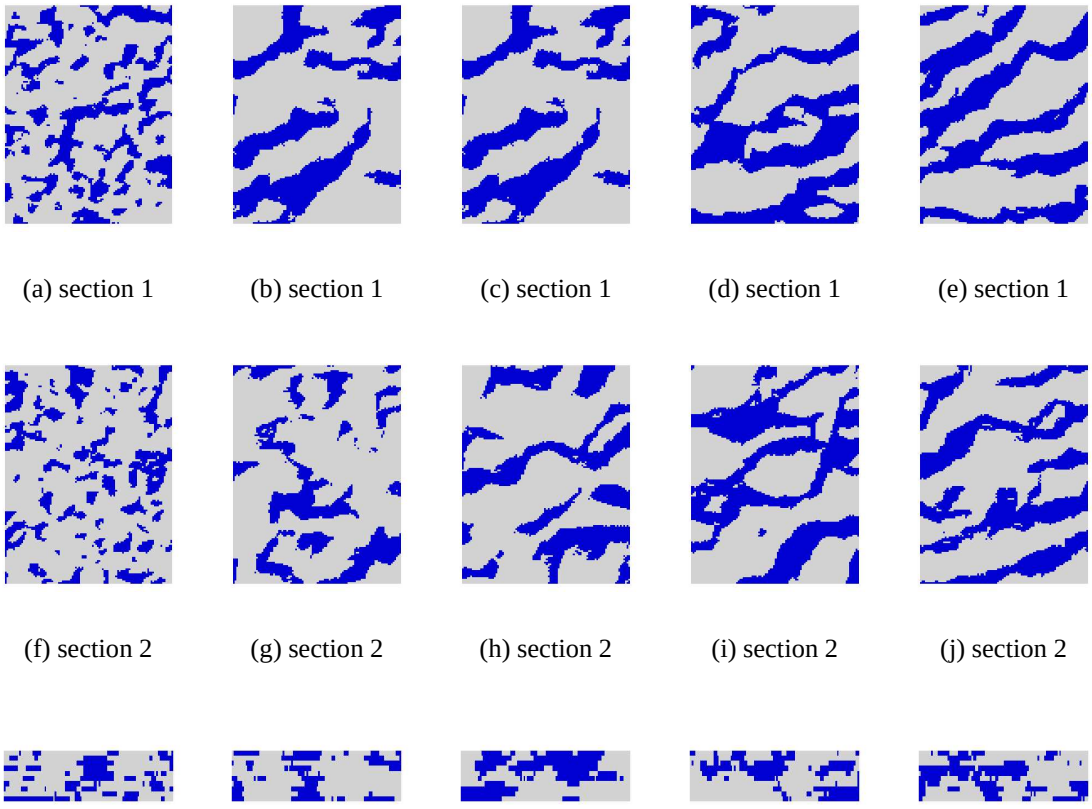


(a) section 1 (b) section 1 (c) section 1 (d) section 1

(e) section 2 (f) section 2 (g) section 2 (h) section 2

(i) XZ section (j) XZ section (k) XZ section (l) XZ section

Figure 3.18: Influence of the template size. Column 1: 30 nodes. Column 2: 60 nodes. Column 3: 100 nodes. Column 4: 200 nodes.

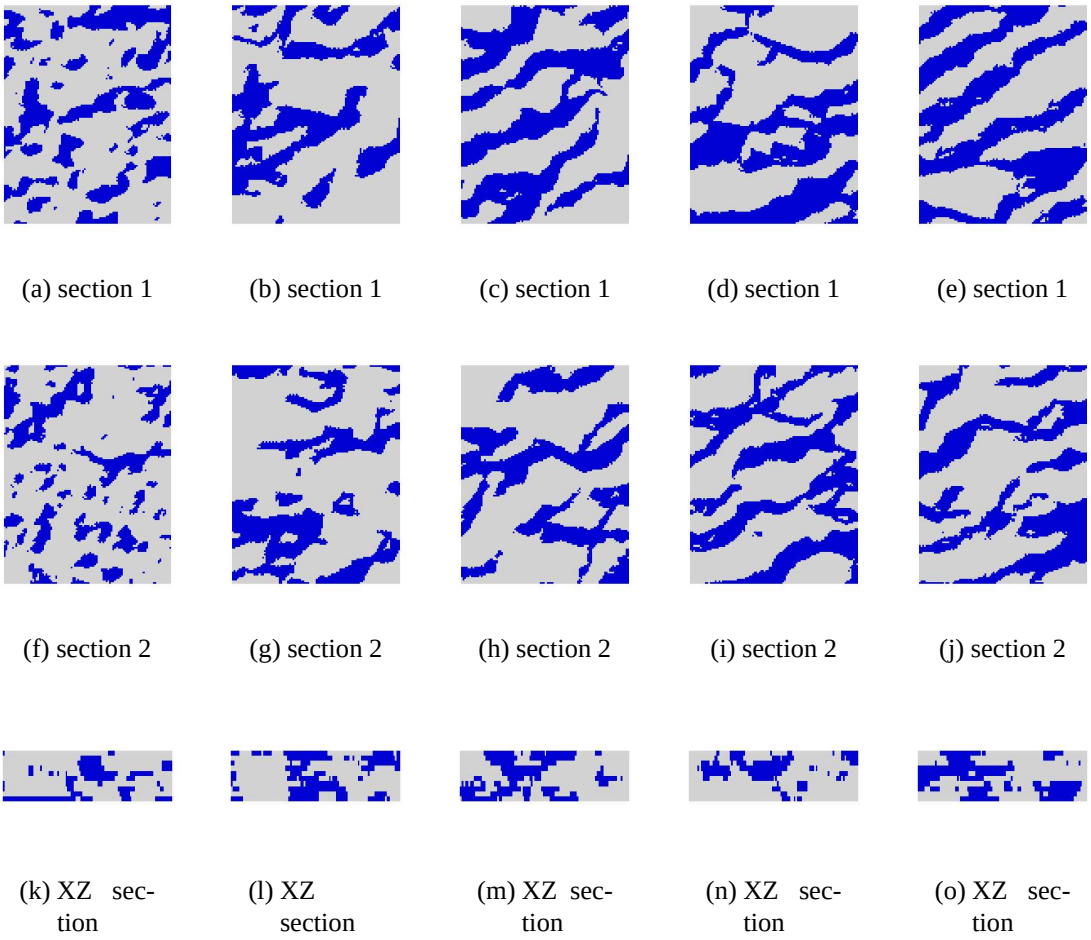


(a) section 1 (b) section 1 (c) section 1 (d) section 1 (e) section 1

(f) section 2 (g) section 2 (h) section 2 (i) section 2 (j) section 2

(k) XZ section (l) XZ section (m) XZ section (n) XZ section (o) XZ section

Figure 3.19: Influence of the number of multiple grids, with a 30-nodes template. Column 1: 1 multiple grid. Column 2: 2 multiple grids. Column 3: 3 multiple grids. Column 4: 4 multiple grids. Column 5: 5 multiple grids.

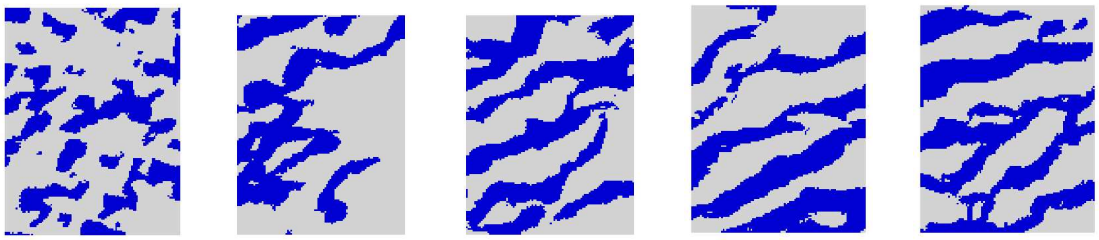


(a) section 1 (b) section 1 (c) section 1 (d) section 1 (e) section 1

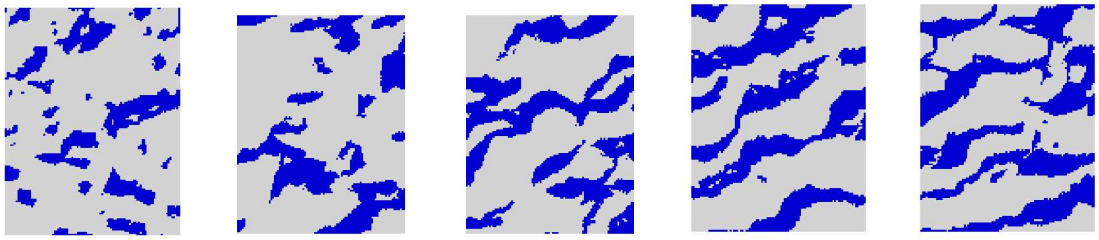
(f) section 2 (g) section 2 (h) section 2 (i) section 2 (j) section 2

(k) XZ section (l) XZ section (m) XZ section (n) XZ section (o) XZ section

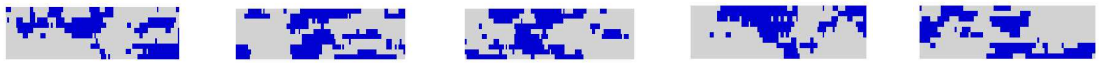
Figure 3.20: Influence of the number of multiple grids, with a 100-nodes template. Column 1: 1 multiple grid. Column 2: 2 multiple grids. Column 3: 3 multiple grids. Column 4: 4 multiple grids. Column 5: 5 multiple grids.



(a) section 1 (b) section 1 (c) section 1 (d) section 1 (e) section 1



(f) section 2 (g) section 2 (h) section 2 (i) section 2 (j) section 2



(k) XZ section (l) XZ section (m) XZ section (n) XZ section (o) XZ section

Figure 3.21: Influence of the number of multiple grids, with a 200-nodes template. Column 1: 1 multiple grid. Column 2: 2 multiple grids. Column 3: 3 multiple grids. Column 4: 4 multiple grids. Column 5: 5 multiple grids.

Parameters description

`PropertySelector_Training:` The grid and property containing the training image

`Nb_Facies:` The number K of different values that the simulated categorical variable Z can take

`Marginal_Cdf:` The marginal pdf of the variable. The probability values must be given in the following order: $P(Z = 0), \ldots, P(Z = K - 1)$

`Max_Cond:` The maximum number J of conditioning data to be retained for each simulation

`Search_Ellipsoid:` The ranges and angles defining the ellipsoid used to search neighboring conditioning data. Template T_J will be automatically built from the search ellipsoid.

`Hard_Data:` The grid and property containing the hard data

`Use_ProbField:` This flag indicates whether the simulation should be conditioned to prior local probability information (e.g. the indicator coding of soft data). Set the flag to 1 to condition to prior local probability information

`ProbField_properties:` The grid and properties containing the prior probability values. One property must be specified for each possible value of $Z(\mathbf{u})$. The k^{th} property corresponds to $P\big(Z(\mathbf{u}) = \mathrm{k} \mid \mathrm{Y}(\mathbf{u})\big)$. This parameter is ignored if `Use_ProbField` is set to 0

`Use_Rotation:` This flag indicates whether to use additional angle information.

`Use_Global_Rotation:` This flag indicates whether to use a global rotation angle. If set to 1, a single angle must be specified: $R(\mathbf{u}) = \mathrm{R}$ is constant for all locations $\mathbf{u}$

`Use_Local_Rotation:` This flag indicates whether to use a local rotation angle: if set to 1, a rotation angle must be specified for each node of the simulation grid. Note that `Use_Global_Rotation` and `Use_Local_Rotation` are mutually exclusive

`Global_Angle:` The global rotation angle. The training image will be rotated by that angle before the simulation begins. This parameter is ignored if `Use_Global_Rotation` is set to 0

`Rotation_property:` The property of the simulation grid containing the local angle category. Angle categories range from 0 to $M-1$ where M is the total number of angle categories. The angles corresponding to each angle category are specified by `Rotation_categories`

`Rotation_categories:` The angles, expressed in degrees, corresponding to each category. The angles are separated by spaces. This parameter is ignored if `Use_Global_Rotation` is set to 0

`Cmin:` The minimal number of training replicates of a given data event that must be found in order for that data event to be retained. This is referred to as n_{min} in step 2(b) of **Algorithm 5**.

`Constraint_Marginal_ADVANCED:` A number in [0,1] indicating the extent to which the marginal pdf of the variable should be matched. See the ω constant in the discussion on marginal distribution reproduction

`Nb_Multigrids_ADVANCED:` The number of multiple grids

`Subgrid_choice:` A flag indicating whether sub-grids should be used. "Sub-grids" is a structured simulation path which possibly allows to use a smaller geometrical template D_J

`Previously_simulated:` The number of nodes belonging to other sub-grids that should be used to simulate a given sub-grid. Typically 4 nodes are used. This parameter is ignored if `Subgrid_choice` is set to 0

Bibliography

Almeida, A. and Journel, A.: 1994, Joint simulation of multiple variables with a markov-type coregionalization model, *Mathematical Geology* **26**(5), 565–588.

Chilès, J. and Delfiner, P.: 1999, *Geostatistics: Modeling spatial uncertainty*, John Wiley & Sons, New York.

Deutsch, C. and Journel, A.: 1992, *GSLIB: Geostatistical Software Library and User's Guide*, Oxford University Press, New York.

Goovaerts, P.: 1994, Comparative performance of indicator algorithms for modeling conditional probability distribution functions, *Mathematical Geology* **26**, 389–411.

Goovaerts, P.: 1997a, *Geostatistics for natural resources evaluation*, Oxford University Press, New York.

Goovaerts, P.: 1997b, *Geostatistics for natural resources evaluation*, Oxford University Press, New York.

Holden, L., Hauge, R., Skare, O. and Skorstad, A.: 1998, Modeling of fluvial reservoirs with object models, *Mathematical Geology* **30**(5), 473–496.

Journel, A.: 1999, Markov models for cross covariances, *Mathematical Geology* **31**.

Journel, A.: 2002, Combining knowledge from diverse sources: An alternative to traditional data independence hypotheses, *Mathematical Geology* **34**(5), 573–596.

Mao, S. and Journel, A.: 1999, Generation of a reference petrophysical/seismic data set: the stanford v reservoir, *Report 12, Stanford Center for Reservoir Forecasting*, Stanford, CA.

Strebelle, S.: 2000, *Sequential simulation drawing structures from training images*, PhD thesis, Stanford University, Stanford, CA.

Tjelmeland, H.: 1996, *Stochastic models in reservoir characterization and Markov random fields for compact objects*, PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Tran, T.: 1994, Improving variogram reproduction on dense simulation grids, *Computers & Geosciences* **20**(7), 1161–1168.

Zhu, H. and Journel, A.: 1993, Formatting and interpreting soft data: Stochastic imaging via the markov-bayes algorithm, **1**, 1–12.