Towards a pattern recognition-based Geostatistics

Jef Caers

Stanford University
Department of Petroleum Engineering

Abstract

Current geostatistical methods can be seen as pattern recognition and classification tools based on two-point statistics, i.e. the variogram. Indicator kriging allows to classify an unknown facies-type or petrophysical property at any location given the facies-type or properties at neighboring sample locations. Indicator simulation amounts to a successive use of this classification tool. In this paper a prospective study is presented that aims at extending the geostatistical toolbox by including better classification tools than kriging, namely neural networks. Kriging, through the variogram, relies only on two-point statistics of the attribute value under study. Geological continuity involves complex patterns that are beyond the capabilities of two-point statistics. Neural network can recognize the multiple point statistics inherent to such complex patterns of continuity. In this paper an iterative simulation technique based on the Metropolis-Hastings sampler is proposed that uses neural networks instead of kriging as a classification tool. It is shown how both hard and soft data, after proper calibration, can be integrated in the resulting simulations.
1 Pattern Recognition

1.1 Introduction

Pattern recognition encompasses a wide range of information processing problems that are of vital importance to a number of fields such as speech recognition, medical diagnosis, the classification of handwritten characters, communication signals and many others (Bishop, 1995). The most efficient tool for solving many pattern recognition problems is the human eye, which perceives seemingly effortlessly, the variability, edginess and directionality of patterns appearing in images existing in sciences and engineering. However, the human eye cannot deal with large repetitive information streams, hence automated recognition devices are needed. The goal of pattern recognition is to classify observed patterns in groups or classes. A set of handwritten symbols 'a' and 'b' could be classified as either 'a' or 'b' by the human eye, or if badly written, might be left open as unclassifiable. The aim of pattern recognition is to design machines (hardware or software) that automate the task of pattern recognition and classification. First attempts at building such machines have relied on understanding why and how humans perform the task of recognition so easily. A marked achievement was made by the introduction of neural networks as pseudo for the human brain, yet their biological/psychological roots were soon left and neural networks became artificial neural networks, which are currently developed as mathematical and statistical techniques.

One important characteristic about man or machine performing pattern recognition is that they have to be trained. Training can be unsupervised or supervised. In an unsupervised training mode, we learn to make a difference from a group of handwritten symbols 'a' and 'b' simply by observing that the set of 'a's looks different from the set of 'b's without knowing that the handwritten 'a' is really an 'a' and the handwritten 'b' is really a 'b'. We discover, unsupervised, groupings or clustering within the set of handwritten symbols (just like blind tasting of wines). A more common mode of training is to work on a collection of examples that are labelled, i.e. the training is supervised. Examples of handwritten symbols are presented that are labelled with the target labels being either 'a' or 'b', i.e. in the training set we know to which class any handwritten symbol belongs. The supervised pattern recognition is then used to classify future examples of newly handwritten symbols 'a' or 'b' into the same set of classes. In both supervised and unsupervised modes features are extracted from the example or object to be recognized and classified. In case of handwritten symbols such features are the locations of a set of black or white pixels as recorded through scanning on a chart.

Pattern recognition can be approached on a purely deterministic basis, devoid from any probabilistic notion of uncertainty; yet for our purposes, we will deal primarily with the field of statistical pattern recognition. In the latter case, the classification of examples or objects from their features is not unique and the object may belong to multiple classes.
More precisely, there is only a probability that the object belongs to any class. For example, the training set of handwritten letters might be particularly bad due to lack of information incurred by an imperfect scanning of the handwritten symbols. This means that on the basis of a training set the pattern classifier may not give a unique result, i.e. newly handwritten symbols cannot be uniquely classified as 'a' or 'b': they are a bit of 'a' and a bit of 'b'. Statistical pattern recognition is particularly relevant to the earth sciences. Measurements are gathered revealing the structural, physical and chemical behavior of rocks at particular locations in space over a domain of observation. Geologists are trained to recognize the essence of these measurements, interpret the data and classify them for future decision purposes. Due to the lack of information, imprecise measurements and the need for calibration of the measurements tools, the resulting classification is not unique and various alternatives exist. No matter how uncertain, the interpretation of the geologist is vital for decision making purposes. This is the area where geostatistics has contributed most, providing a framework that can merge geological interpretation and actual measurements.

The intent of this paper is to open a new avenue which could be termed ”pattern recognition-based geostatistics”. Current geostatistical methods already rely on pattern recognition but limited to two-point statistics. A typical geostatistical study will infer variogram models (a two-point statistic) from the measurements (data), including variograms of data of the same type (direct variograms) and data of different types (cross-variograms). Then, the task of simulation is to reproduce that variogram model conditional to the available local data.

Variograms can only capture two-point spatial continuity which is too limited to describe the true underlying curvi-linear geological patterns of continuity. The limitation to two-point information limitation is counter-intuitive with the pattern recognition paradigm. Human eyes build patterns not by taking information two by two at a time, but construct/recognize an image by looking at information at multiple locations at the same time. A more advanced classification tool should therefore consider all data (all features) at the same time and make a classification on such basis. Such information is termed multiple-point information or in terms of statistical pattern recognition: multiple-point statistics.

In the next section we will elaborate further on this key concept and show that using modern, well understood, pattern recognition techniques, the current geostatistical toolbox can be extended.

1.2 Pattern recognition based on two-point statistics

Indicator methods have been widely used in geostatistics for reservoir facies characterization (categorical variable case) and for simulation of petrophysical properties (continuous
variable case). Indicator kriging allows determining the probability that a certain facies occurs at location \( \mathbf{u} \) given that any other facies occur at neighboring locations \( \mathbf{u}_\alpha \). Indicator simulation amounts to a sequential use of probabilistic classification based on indicator kriging.

Indicator kriging can be viewed as a statistical pattern recognition and classification tool. The object to be classified is the facies or petrophysical value at an unsampled location \( \mathbf{u} \) (the object), and the data available for that classification (the features) are the various facies or petrophysical values observed at neighboring sample locations \( \mathbf{u}_\alpha \). The classification done by kriging relies on a covariance model between features (data-to-data covariance) measuring the redundancy of information and a covariance model between the features and the unknown object (data-to-unknown). The classification given by kriging is not unique and the object may belong to various categories. The main result of an indicator kriging is therefore not an estimated indicator but the probability that a particular facies at location \( \mathbf{u} \) prevails given the information available at neighboring locations. Hence the estimated indicator is interpreted a conditional probabilty of having a particular facies at location \( \mathbf{u} \). On the basis of these probabilities a final decision of classification can be made.

Sequential indicator simulation is a series of successive classifications or drawings from such conditional probability distributions, each classification/drawing depending on classifications/drawings previously made at neighboring locations. The ultimate goal of simulation is to obtain through the successive classifications, a reproduction of the trained patterns into the reservoir models conditioned to the sample information available. Since indicator kriging is a classification tool that is built on two-point statistics only, pattern reproduction with indicator simulation will only reproduce the two-point statistics of patterns. The drawing from a conditional distribution is a key concept in geostatistics. The conditional probability is not a goal on itself, only variogram reproduction is; the conditional probability is constructed through indicator kriging using the variogram, hence drawing from it ensures variogram reproduction (Journel, 1994).

Current indicator simulation methods rely only on two-point statistics: the indicator variograms, although theoretical expansions to multiple-point statistics have been formulated (Journel and Alabert, 1989; Guardiano and Srivastava, 1992). Indicator variograms are certainly an improvement in terms of connecting specific classes of values when compared to the classical direct variogram. Yet, they can only capture two-point spatial continuity which is too limited to describe the true underlying curvi-linear geological patterns of continuity. When estimating an indicator variogram or any variogram for that matter, we rely on the recognition of geological structure through the pooling of two-point information. Recognizing geologically interpretable structure in the experimental variogram values leads to a model variogram. The classification made by indicator kriging therefore relies only on the two-point dependency/correlation between each datum (each single feature) and the unknown facies or petrophysical value.
In this paper, an indicator simulation is proposed that relies on classification performed by feed-forward neural networks instead of indicator kriging. Feed-forward neural network have the capability to make classification by considering all data (all features) at the same time, hence taking into account multiple-point information. As in any pattern recognition problem, neural networks need to be trained to recognize complex geological patterns of continuity and to be able to classify unsampled locations on the basis of neighboring measurements. Such training information is depicted in training images.

1.3 Pattern recognition based on diverse data types

There is a clear need for a new generation of sequential simulation methods that allow integration of diverse data with varying support size and resolutions (scales). The goal is to be able to estimate or simulate a facies type or a petrophysical value at any unsampled location \( \mathbf{u} \), given all possible direct information (hard measurements) and indirect information (soft measurements). The indirect information (seismic/flow) has typically a complex, non-linear relation with the hard information. For example, a set of porosity values in a particular area or block of the reservoir is non-linearly related to the seismic impedance measured on that whole block. Furthermore, that relation is multiple-point, i.e. the seismic impedance relates to all porosity taken all together, it is not a simplistic single point combination (such as an average) of the porosity values taken one at a time. That relation determines the information content on the hard variable provided by indirect measurements and must be calibrated. Such data calibration relates to the physics of the measurement (wave propagation or flow process) and the calibration must take into account the multiple-point relation existing between hard and soft variables.

Surveying the present geostatistical literature on data integration we observe a shortage of methods that allow a physics-based integration of diverse data types.

- Traditional sequential simulation methods only allow conditioning to linear or pseudo non-linear single point averages.

- Simulated annealing is a Markov chain Monte Carlo method that allows integration of multiple-point information through an objective function, yet it is too slow as a practical method for data integration in large grid-size reservoirs. Furthermore, the implicit random function generated by simulated annealing depends on the type of distribution of starting images and on the parameters defining the cooling schedule. In this paper, we will rely on methods whose random functions are proven to be independent of the starting images and are sure to converge (at least in the limit).

- Inverse methods are particularly difficult when using a pixel-based approach. Inverse modelling cannot handle the millions of pixel values needed to represent a real 3D reservoir. Inverse modelling is typically associated with a low parameter space (for
example objects) and are usually limited by multivariate Gaussian assumptions (Tarantola, 1987).

Pattern recognition and classification can be extended as a tool for data integration. The pattern recognition and classification tools should be extended to include information (indirect features) that relates indirectly to an unknown facies type or petrophysical property to be classified. As for any pattern recognition tool, training data must be provided to calibrate hard and soft information. To obtain such calibration information, the physical process (seismic/flow) determining the soft information is simulated on the hard-data training image to obtain a soft-data training image (Figure 1). Pattern recognition and classification is then used to calibrate the information content of soft data using both the hard-data training image and the soft data training image. The intent is not only to reproduce the spatial patterns of both hard and soft data but also the physics relating hard and soft data. Current geostatistical methods based on two-point statistics only reproduce the two-point correlation between hard and soft data not the structure of the soft information. The goal is to generate hard attribute images which would match the soft data were they processed by a seismic/flow simulator.

2 Methodology outline

Consider first the categorical case with $K$ categories $s_k, k = 1, \ldots, K$. The random function model $S(u)$ describes the facies type at each location $u$ in the field of study. The indicator random variable $I(u; s_k)$ describes each class $s_k$ of the random function $S(u)$

$$I(u; s_k) = \begin{cases} 1 & \text{if } S(u) \text{ belongs to } s_k \\ 0 & \text{else} \end{cases}$$

The strength of the indicator paradigm is that it maps the problem of estimating probabilities to the problem of estimating expectations

$$\Pr\{I(u; s_k) = 1\} = E[I(u; s_k)]$$

and, similarly for the conditional expectation:

$$\Pr\{I(u; s_k) = 1 | (n)\} = E[I(u; s_k)|(n)]$$

(1)

where $(n)$ is the collection of $n$ data $S(u_\alpha) = s_{k\alpha}, \alpha = 1, \ldots, n$ with $s_{k\alpha}$ the category prevailing at each location $u_\alpha$. At each datum location we therefore have a vector of size $K$ of indicator data values $I(u_\alpha; s_k), k = 1, \ldots, K$. Expectations are best determined using least squares regression methods, i.e. by minimizing an error variance. The best estimator
in a least square sense of the indicator variable $I(u; s_k)$ is precisely (i.e. that has minimim error variance) the conditional expectation (1). This means that conditional probabilities can be determined by least square estimation of the unknown indicator values.

Prior to any indicator kriging, the prior probability of category $s_k$ prevailing at location $u$ is known and denoted as

$$\Pr\{I(u; s_k) = 1\} = \mathbb{E}[I(u; s_k)] = p_k$$

Indicator kriging takes a linear combination of the indicator data $I(u; s_k)$ and allows an updating of that prior probability $p_k$. The conditional probability

$$\Pr\{I(u; s_k) = 1|(n)\} = \mathbb{E}[I(u; s_k)|(n)]$$

is estimated by

$$i^*(u; s_k) = \sum_{\alpha=1}^{n} \lambda_\alpha(u, s_k)i(u; s_k) + [1 - \sum_{\alpha=1}^{n} \lambda_\alpha(u, s_k)]p_k$$

Indicator kriging determines the conditional probabilities $\Pr\{I(u; s_k) = 1|(n)\}$ that the unknown facies data at location $u$ belongs to category $s_k$ given the facies at neighboring locations $u_\alpha$, $\alpha = 1, \ldots, n$. The weights $\lambda_\alpha(u, s_k)$ are determined using the indicator variograms (Goovaerts, p. 294).

Indicator kriging can be viewed as a classification tool because it determines the probability for the unknown facies value at $u$ to be in any of the class $s_k, k = 1, \ldots, K$.

The kriging expression (2) can be rewritten as

$$i^*(u; s_k) - p_k = \sum_{\alpha=1}^{n} \lambda_\alpha(u, s_k)[i(u_{\alpha}; s_k) - p_k]$$

The conditional probabilities $\Pr\{I(u; s_k) = 1|(n)\}$, estimated by $i^*(u; s_k)$ can then be interpreted as posterior probabilities because they update the prior probability $p_k$ based on known prior updating values $[i(u_{\alpha}; s_k) - p_k]$ at the data locations.

The limitation of kriging as a classification tool lies in the linear representation of (2). Only two-point statistics (variograms) are used to determine the weights $\lambda_\alpha$ and thus the posterior probabilities used for classification. Preferably, in the most general case, we would like to obtain a non-linear relation between posterior probabilities and all $(n)$ data considered jointly:

$$\Pr\{I(u; s_k) = 1|(n)\} = \phi((n); \theta; s_k)$$

with $\theta$ being a set of parameters for the function $\phi$. The function $\phi$ is a general non-linear function that determines the posterior probabilities for each class $s_k$. The superiority of the general representation $\phi$ over the kriging representation (2) lies in the fact that $\phi$
determines the posterior probabilities of all data \( n \) considered jointly instead of weighting them one by one as done in kriging. Such joint consideration of the data coincides with the pattern recognition paradigm that patterns are constructed and hence should be recognized on the basis of multiple point information instead of a collection of single point events. In the next section, feed forward neural networks are proposed to model the function \( \phi \) in (3) with a least squares method to determine the weights \( \theta \).

As in any pattern recognition problem the function \( \phi \) needs to be trained using data collected from the training image. We will use supervised training. Data is collected from the training image using a template \( t \) that defines a set of locations in the neighborhood of any location \( u \) in the training image:

\[
u_\alpha = u + h_\alpha \quad \alpha = 1, \ldots, n_t
\]

\( n_t \) the number of locations within the template \( t \). Figure 2 shows how templates are defined and used to scan a training image. The template geometry is defined by the set of \( n_t \) vectors \( \{h_\alpha, \alpha = 1, \ldots, n_t\} \). A specific template realization \( t(u) \) comprises both its geometry and the \((n_t + 1)\) attribute values

\[
t(u) = \{s(u); s(u + h_\alpha), \quad \alpha = 1, \ldots, n_t\}
\]

The set of neighborhood template values is represented by the vector

\[
s_t(u) = \{s(u + h_\alpha), \quad \alpha = 1, \ldots, n_t\}
\]

hence the template realization \( t(u) \) can be written as:

\[
t(u) = \{s(u); s_t(u)\}
\]

In pattern recognition language, the central location with value \( s(u) \) is denoted as target label or class of the template realization, whereas the set of neighboring values \( \{s(u + h_\alpha), \quad \alpha = 1, \ldots, n_t\} \) are the features. The function \( \phi \) is then function of the set of features

\[
\phi = \phi(s_t(u); \theta; s_k)
\]

Scanning the training image provides a set of template realizations

\[
\{t(u_j), u_j \in A\}
\]

\( A \) is the training image such that \( u_j \in A \) and \( u_j + h_\alpha \in A, \forall \alpha = 1, \ldots, n \). Alternatively we can consider the training image \( A \) eroded by the template geometry

\[
A_{\{h_\alpha\}} = \{u \in A \ s.t. \ u + h_\alpha \in A, \forall \alpha = 1, \ldots, n_t\}
\]

The training image provides the set

\[
\{t(u), u \in A_{\{h_\alpha\}}\}
\]
Pattern recognition and classification is not a goal on itself, the trained pattern need to be reproduced using a simulation method. Contrary to classification using indicator kriging based on two-point statistics, the intent is to reproduce multiple-point information using the function $\phi$ as a classification tool instead of kriging. Also, instead of using classical sequential simulation, a Markov chain Monte Carlo simulation (Metropolis et al., 1953; Ripley, 1977; Hegstad et al., 1993) is needed, the reason for this is discussed in later sections. The latter is also termed iterative sequential simulation a term coined independently in geostatistics (Srivastava, 1992). In classical sequential simulation, one visits each node of the simulation grid using a random path, at each location $\mathbf{u}$, the $n$ closest original data values and previously simulated nodes are searched. Using this set of neighboring values, a local conditional distribution is constructed and a simulated value drawn. Then one proceeds to the next node. Thus, at each node to be simulated the conditional distribution has to be built anew since the configuration of neighboring data values changes from one node to another. When all nodes of the grid have been simulated, simulation is stopped.

The motivation for using iterative sequential simulation comes from the speed advantage of having a fixed data geometry at each data location. Similarly, the neural net training of the function $\phi$ is done using a fixed template, i.e. with a fixed data geometry. The simulation grid is initialized with random values (or values drawn randomly from a target histogram). Data values are assigned at the data locations. Instead of visiting each node only once, the simulation algorithm cycles multiple times over the whole simulation grid, each time updating the value at the visited node. Conditioning data locations are never visited. The details of the updating process are discussed in the next sections.

The general algorithm outline proposed is as follows:

1. **Pattern recognition and classification**

   (a) Define a template $t(\mathbf{u})$ with which the training image $A$ is scanned to construct the dataset

   \[
   \{s(\mathbf{u}_j); s(\mathbf{u}_j + \mathbf{h}_\alpha), \quad \alpha = 1, \ldots, n_t \}, \quad j = 1, \ldots, N, \quad i.e. \quad \forall \mathbf{u}_j, \mathbf{u}_j + \mathbf{h}_\alpha \in A
   \]  

   (b) Define a classification function $\phi$ of type (4) with parameters $\mathbf{\theta}$ as a specific joint function of all $n_t$ values $s(\mathbf{u} + \mathbf{h}_\alpha), \quad \alpha = 1, \ldots, n_t$

   (c) Train the function $\phi$ to determine a posterior probability of type (2), that is the probability of obtaining $S(\mathbf{u}) = s_k$ given any set of $n_t$ data \{s(\mathbf{u} + \mathbf{h}_\alpha), \quad \alpha = 1, \ldots, n_t\}. Training the function $\phi$ amounts to determine its parameters $\mathbf{\theta}$.
2. Pattern reproduction using simulation

(a) Initialize the simulation by freezing the original data values at their locations and filling-in the remainder nodes with values drawn from the global target histogram (prior class probabilities)

(b) Define a random path visiting all non-frozen nodes

(c) Start loop : visit all nodes, at each node do

- Update the class-value at that node using the function $\phi$

(d) Stop loop when convergence is reached, if not goto (c)

(e) If another image is required start all over from (a)

In the next sections we go into every detail of each step of this algorithm.
3 Pattern recognition with feed-forward neural network

3.1 Introduction

Feed-forward neural networks are known as a powerful tool in pattern recognition (Bishop, 1995). Most applications of neural networks deal with so-called hard classification, i.e. after proper training, the neural network is able to uniquely define the class to which a certain object belongs. In the following application of neural networks, we will rely more on the neural network capability to perform soft-classification. In such case, very rarely can a particular object be uniquely classified, usually a number of classes are possible for classification. Each of these possible classes is assigned a certain probability. Such probabilities are the posterior or conditional probabilities of the type (2).

A neural network, similar to kriging, defines a mapping between any number of input variables and any number of output variables. The particularity of neural networks is that they start with linear combinations of the input variables, then process those combinations through non-linear functions, last they linearly recombine these non-linear outputs to define the output variables. We will first introduce the general architecture of neural networks, then define the error function that is used to train such neural networks on the training dataset (5) provided by scanning the training image.

3.2 Network and error function

A multiple layer feed-forward network with single hidden layer is shown in Figure 3. The network has an input layer, a hidden layer and an output layer. The number of input and outputs are defined by the problem at hand, the number of hidden layers retained depends on the information contained in the data. In mathematical terms the mapping can be written as follows

\[ y_k(x, \theta) = \sum_{\beta=1}^{m} o_{\beta k} \left( \sum_{\alpha=1}^{n_t} w_{\beta \alpha} x_\alpha \right), \quad k = 1, \ldots, K \]

\[ = \phi(x_1, \ldots, x_{n_t}; \theta; k) \]

where \( x = \{x_1, \ldots, x_{n_t}\} \) is the set of input variables and \( y = \{y_1, \ldots, y_K\} \) the set of output variables. \( m \) is the number of hidden layer nodes. \( m = 3, K = 3, n_t = 4 \) in Figure 3.

\( \theta = \{o, w\} \) is the set of parameters to be determined, with

\[ w = w_{\beta \alpha}; \beta = 1, \ldots, m; \alpha = 1, \ldots, n_t \]
and \[ \mathbf{O} = \mathbf{o}_{\beta k}, \beta = 1, \ldots, m; k = 1, \ldots, K \]
The function $T$ is a non-linear function, typically sigmoidal. For $T$ we will take the following function throughout this paper:

\[ T(z) = \frac{1}{1 + \exp(-z)} \]

$T(z)$ is termed the logistic sigmoid activation function. The logistic function maps the interval $[-\infty, +\infty]$ onto $(0, 1)$. If $|z|$ is small then $T(z)$ is approximately linear, hence in a sense a network with $T(z)$ contains the linear network (i.e. a linear relation between input and output) as a special case. Note also that

\[ \frac{dT}{dz} = T(z)(1 - T(z)) \]
a property that will prove useful in training the neural network.

A set of data is needed to train the neural network (determine the parameters $\mathbf{\theta}$) in the supervised mode. The data comprises $N$ sets of input (features) and corresponding target output (known category for that set of features)

\[ \{(x^i_1, \ldots, x^i_n); (t^i_1, \ldots, t^i_K)\}, \quad j = 1, \ldots, N \]

The $t^j_1, \ldots, t^j_K$ are the indicators quantifying to what category the set of features $x^j_1, \ldots, x^j_n$ belongs to. To train a neural network a training criterion or objective function must be provided which minimizes the difference between the network outputs $y_k(\mathbf{x}, \mathbf{\theta})$ and the target outputs $t_k$ as provided by the set of data. Such criterion could be a sum-of-squares error function

\[ \mathbf{\theta} \text{ determined such that } Ef = \frac{1}{2} \sum_{j=1}^{N} \sum_{k=1}^{K} (y_k(\mathbf{x}^j, \mathbf{\theta}) - t^j_k)^2 \text{ is minimum} \quad (6) \]

This function needs to be minimized to determine the weights $\mathbf{\theta}$.

Using the training image and a template, a set of input and target output data is determined by scanning training image(s). The set of input data $\mathbf{x}$ are facies-types within the neighborhood template of any location $\mathbf{u}_j$

\[ \mathbf{s}_i(\mathbf{u}_j) = \{s(\mathbf{u}_j + \mathbf{h}_a), \quad \alpha = 1, \ldots, n_t\}, \quad j = 1, \ldots, N \quad (7) \]

The target data $t_k$ are the indicators of the facies type of these central locations $\mathbf{u}_j$

\[ \{i(\mathbf{u}_j; s_k), \quad k = 1, \ldots, K\}, \quad j = 1, \ldots, N \quad (8) \]
The mathematical expression for the neural network relating input and output is written:

\[
y_k(s_i(u), \theta) = \sum_{\beta=1}^{m} o_{\beta k} T \left( \sum_{\alpha=1}^{n_i} w_{\beta \alpha} s(u + h_{\alpha}) \right), \quad k = 1, \ldots, K \tag{9}
\]

\[
= \phi(s(u + h_1), \ldots, s(u + h_{n_i}); \theta; k)
\]

As in indicator kriging, the key idea when training the network is not to identify the target indicators (8) but to determine the posterior probabilities of the class membership \(\Pr\{I(u; s_k) = 1|s_i(u)\}\). Three important questions must be addressed if we want to interpret the neural network outputs as probabilities:

1. Is the output (9) of the network always in the interval \([0, 1]\)?
2. Do the network outputs sum to one?
3. Is the output of the network an expectation? If so, that expectation is taken over an indicator, hence it is a probability, see relation (1).

Questions 1 and 2 are similar to the order relations that indicator kriging estimates must meet.

Consider the third condition under the limit case of an infinitely large dataset obtained by scanning an infinitely large image. In such limit the sum of squares (6) becomes

\[
Ef = \lim_{N \to \infty} \frac{1}{2N} \sum_{k=1}^{K} \sum_{j=1}^{N} (y_k(s_i(u_j), \theta) - i(u_j; s_k))^2
\]

\[
= \frac{1}{2} \sum_{k=1}^{K} \mathbb{E}[(y_k(S(u_j), \theta) - I(u_j; s_k))^2]
\]

where \(N\) is introduced in the denominator to allow for the limit to exist. In Appendix 1 it is shown that in such limit case the minimum of the error function \(Ef\) is found by the following solution

\[
y_k(s_i(u_j), \hat{\theta}) = \mathbb{E}[I(u_j; s_k)|s_i(u_j)], \quad k = 1, \ldots, K
\]

where \(\hat{\theta}\) are the set of weights that minimize the error function \(Ef\). This means that if \(\hat{\theta}\) is found for which \(Ef\) is minimal, then the output of network can thus be interpreted as a conditional expectation of the indicators at any location \(u_j\) given its neighbor class values \(s_i(u_j)\). As the expectation of an indicator is a probability, neural networks trained with indicators as target output data \(t_k\) yield conditional probabilities as output values \(y_k\). Again, this result is similar to indicator kriging.

If the network outputs represent probabilities then they should sum to unity. It can be shown (Bishop, 1995, p 200-201) that the outputs \(y_k\) of a feed-forward neural networks,
trained with a sum-of-squares error function will satisfy any linear constraint that the target output data \( t_k \) satisfies. The target data for the network output are the set of indicators (8) which sum to unity, hence the probabilities will also sum to unity. There is however no guarantee yet that the probabilities are within \((0, 1)\).

The network output can be constrained to the \([0, 1]\) interval in two possible ways

- The output probabilities can be scaled to be within the interval \([0, 1]\) for any arbitrary input (posterior correction).
- The network architecture can be constructed such that the outputs are guaranteed to be in the \([0, 1]\) interval and sum to unity. This is achieved by using a so-called softmax condition for the network outputs (Bridle, 1990)

\[
y_k = \frac{\exp(a_k)}{\sum_{k'=1}^{K} \exp(a_{k'})}
\]

where

\[
a_k = \sum_{\beta=1}^{m} o_{\beta k} T \left( \sum_{\alpha=1}^{n} w_{\beta \alpha} x_{\alpha} \right)
\]

Instead of the usual postprocessing to correct order relation problems in indicator kriging, the neural network is built to guarantee all order relations (prior correction).

The network output should satisfy a last condition. As they are interpreted as posterior probabilities, the average of each network output calculated from all input data should approximately correspond to the prior class probabilities, i.e.

\[
\Pr\{I(\mathbf{u}; s_k) = 1\} \approx \frac{1}{N} \sum_{j=1}^{N} \Pr\{I(\mathbf{u}; s_k) = 1|s_j(\mathbf{u})\}, \; \forall k
\]

The left-hand side is known from the data, the right-hand side can be calculated when the net is trained. Hence, this represents a check that the neural network has been properly trained.

3.3 Minimizing the error function

Validation

The reasons for minimizing the error function (6) is to determine the weights \( \theta \) of the neural network. The trained weights can then be used to evaluate the output of any given input, i.e. do prediction. Hence these weights will most likely be applied to input data
vectors different from any of the training input data vector. Thus, one needs to ensure that the trained weights also work on an input vector that is not in the training set. In order to ensure that the training is done properly and that the minimization of the error function does not lead to an overfitting of the target data, a validation set can be considered

\[ \{ s(u_j); s^v(u_j; s_k) \}, \quad j = 1, \ldots, N^v \]

At each step of the training, in addition to the training error, a validation error is calculated as

\[ E_f^v = \frac{1}{2N^v} \sum_{k=1}^{K} \sum_{j=1}^{N^v} (y_k(s(u_j), \theta) - s^v(u_j; s_k))^2 \]  \hspace{1cm} (11)

Once the neural network starts overfitting the training data, the validation error will increase and the training is stopped. Validation data can be obtained from a second training image or by splitting the training set of the original image in two parts: one part for the training data and the other for the validation data. That split can be random or structured, e.g. the top part of the training image is validation data, the bottom part is training data.

**Code**

A GSLIB-standardized FORTRAN90 code **bpnn** has been developed to minimize the error function under the constraint given by the softmax condition (10). Unfortunately, all public domain code for neural network training with back-propagation are written in C/C++ or designed with heavy GUI, usually platform-dependent. The code developed here is easily accessible and can be integrated into any GSLIB-style program. The essentials of back-propagation are not discussed as most text-books on neural network cover this topic in great detail (Bishop, 1995; Ripley, 1996).

The minimization is performed using a quasi-Newton method as coded in Press et al. (1996, p. 420-422).
4 Pattern reproduction with Markov chains

Traditional sequential simulation methods rely on the sequence of conditional distributions of nodal values given the original sample data and previously simulated values. The method is called non-iterative because each node is visited only once. The conditional distributions are typically constructed from some form of kriging. The data configuration for each node to be simulated changes from location to location and from one realization to another. Including multiple-point statistics in the traditional methodology is difficult because of that ever changing data event with both changing geometry and changing data values. Solutions for dealing with complex data events in traditional sequential simulation are proposed in Strebelle (1999). In the pattern recognition approach here proposed for simulation, the geometry of data used to construct the conditional distribution does not change and is provided through a template $t$.

The simulation grid is initially allocated with random values drawn from a prior distribution (e.g. the target histogram). We then perform an iterative sequential simulation or Markov chain Monte Carlo simulation. More precisely, we loop through the complete image a number of times, updating each visited node, until the image reflects the desired structure.

4.1 Markov Chain Monte Carlo Simulation

In a paper included in this SCRF report the relevant theory for applying Markov chain Monte Carlo methods for spatial simulation is reviewed. The key idea when using Markov chains is to sample from a unique multivariate distribution. That multivariate distribution can be explicit (it is known), or it can be implicit in which case it is unknown yet has some known properties, such as a given variogram, or better, a given multiple-point statistic. To obtain realizations of this explicit or implicit multivariate distribution one starts with a distribution of initial images, for example the set of random images drawn from a given histogram. Starting with any one initial image, the Markov chain updates this image iteratively until some form of convergence is obtained. Typically, at each step in the iteration of the chain only one nodal value is changed. This means that, in order to determine the next step in the chain, we only need to know what the image is at the current step, we do not need to know what happened at any of the previous steps. The latter is the Markov property of the chain, i.e. the image generated next by the updating mechanism depends only on the current image.

To generate a new realization, the Markov chain starts all over again from a new initial image. For practical use, Markov chains should have two important properties

- The set of resulting realizations from the Markov chain should not depend on the distribution of images that initiate the chain. In mathematical terms: the multivariate
distribution we generate should not depend on the initial distribution. Otherwise, we have to specify for each set of multiple realizations, what the initial images were.

- The chain should always draw from the same multivariate distribution, or in layman terms, the set of realizations we generate using the chain should not depend on the updating mechanism we use. For example, in the case of drawing from a multivariate Gaussian distribution, the updating mechanism need not feature any Gaussian property! In the case of an implicit random function with known properties, we should have an updating mechanism that generates these properties and yet always draws from the same random function independent of what that updating mechanism is. In mathematical terms: convergence should be ensured.

One of the most well known example of Markov chain sampling in geostatistics is simulated annealing. Simulated annealing can be shown to satisfy both of these two properties under very specific conditions of the cooling schedule (Aarts and Korst, 1989). These conditions also depend on the type of objective function used, hence for each new problem, a very specific cooling schedule needs to be designed. In most practical cases this is seldomly done.

In the next section, we use a Metropolis-Hastings sampler instead of annealing as an updating mechanism. For the Metropolis-Hastings sampler it is shown that the above mentioned properties hold: independence of the initial set and ensured convergence.

### 4.2 The Metropolis-Hastings sampler

In this paper, iterative simulation with the Metropolis-Hastings sampler is proposed. For a general description of the Metropolis-Hastings sampler, we refer to the theory paper in this SCRF report. In this section we show how the sampler is used to perform stochastic simulation.

Typically (but not necessarily), the image being constructed is updated iteratively one pixel at a time. The Metropolis-Hastings sampler has an updating mechanism that is twofold: a proposal for a new pixel value and an acceptance probability for that proposal. The current and new image differ only at the single location where the possible switch/change is proposed.

- **Proposal**: a new image is proposed. For that purpose one selects a random location \( u \) in the current image and proposes to change the current facies category \( s_{\text{old}}(u) \) into a new facies category \( s_{\text{new}}(u) \). For example, \( s_{\text{new}}(u) \) could be drawn randomly from a prior distribution (histogram). The probability of proposing the new image is described by a so called transition matrix with elements

\[
Q(s_i, s_j) \quad i = 1, \ldots, K; j = 1, \ldots, K
\]
describing the probability of proposing a new value \( s_{k^\text{new}}(u) = s_j \) at \( u \) given the current value \( s_{k^\text{old}}(u) = s_i \) at \( u \). For example, when the new pixel value is drawn randomly from a prior distribution:

\[
Q(s_{k^\text{old}}, s_{k^\text{new}}) = \frac{1}{N} \Pr\{S(u) = s_{k^\text{new}}\} = \frac{1}{N} p_{k^\text{new}}
\]

with \( p_{k^\text{new}} \) the prior probability for \( s_{k^\text{new}} \) to be drawn. \( 1/N \) is the probability of selecting a particular node in the grid of \( N \) nodes. Note that in this particular case \( Q \) is independent of location \( u \) and the current facies type \( s_{k^\text{old}} \).

- **Probability of acceptance:** this is the probability that the proposed new image be accepted, given the current image. The probability is denoted by \( \alpha(s_i, s_j) \). The Metropolis-Hastings sampler dictates that \( \alpha(s_i, s_j) \) should be as follows

\[
\alpha(s_{k^\text{old}}(u), s_{k^\text{new}}(u)) = \min \left\{ 1, \frac{\Pr\{\text{changing location } u \text{ to } s_{k^\text{new}}\} Q(s_{k^\text{new}}, s_{k^\text{old}})}{\Pr\{\text{keeping } s_{k^\text{old}} \text{ at location } u\} Q(s_{k^\text{old}}, s_{k^\text{new}})} \right\}
\]

For the particular proposal (12), this is written:

\[
\alpha(s_{k^\text{old}}(u), s_{k^\text{new}}(u)) = \min \left\{ 1, \frac{\Pr\{\text{changing location } u \text{ to } s_{k^\text{new}}\} p_{k^\text{old}}}{\Pr\{\text{keeping } s_{k^\text{old}} \text{ at location } u\} p_{k^\text{new}}} \right\}
\]

Note that the Metropolis-Hastings sampler defines a total transition probability \( P \) to go from the current image with value \( s_{k^\text{old}}(u) \) to the new image with value \( s_{k^\text{new}}(u) \) as the combination of proposal and acceptance probability:

\[
P(s_{k^\text{old}}(u), s_{k^\text{new}}(u)) = Q(s_{k^\text{old}}, s_{k^\text{new}}) \times \alpha(s_{k^\text{old}}(u), s_{k^\text{new}}(u))
\]

The probability of changing that current class value to a new class value depends on the value of the neighboring class values in the template centered at \( u \). Using a trained neural network with parameters \( \hat{\theta} \), one can assess for the current class value \( s_{k^\text{old}}(u) \) the probability of having that class value at location \( u \) given the \( n_t \) neighboring class values \( s_i(u) \) within the template at locations \( u + h_a, a = 1, \ldots, n_t \) as

\[
\Pr\{\text{keeping } s_{k^\text{old}} \text{ at location } u\} = \Pr\{I(u; s_{k^\text{old}}) = 1|s_i(u)\} = \phi(s_i(u), \hat{\theta}, s_{k^\text{old}})
\]

Likewise, for the proposed new value \( s_{k^\text{new}}(u) \), using a trained neural network one can assess its probability

\[
\Pr\{\text{changing location } u \text{ to } s_{k^\text{new}}\} = \Pr\{I(u; s_{k^\text{new}}) = 1|s_i(u)\} = \phi(s_i(u), \hat{\theta}, s_{k^\text{new}})
\]

The Metropolis-Hastings acceptance criterion considers the ratio of probabilities of these two events and defines the probability of switching from the current value to the new value as
\begin{equation}
\alpha(s_{k^\text{old}}(\mathbf{u}), s_{k^\text{new}}(\mathbf{u})) = \min \left\{ 1, \frac{E[I(\mathbf{u}; s_{k^\text{new}})]}{E[I(\mathbf{u}; s_{k^\text{old}})]} \right\} \frac{p_{k^\text{old}}}{p_{k^\text{new}}} \right\} 
= \min \left\{ 1, \frac{\phi(s_t(\mathbf{u}), \theta, s_{k^\text{new}})}{\phi(s_t(\mathbf{u}), \theta, s_{k^\text{old}})} \frac{p_{k^\text{old}}}{p_{k^\text{new}}} \right\}
\end{equation}

4.3 Convergence

An important issue that needs to be addressed is when to stop the iteration of the Markov chain. For the Metropolis-Hastings sampler the convergence is ensured, yet it is not known exactly when. Two methods are proposed that deal with the issue of convergence:

- The chain is monitored: at a regular interval important statistics (variograms, multiple point statistics) are calculated from the image and checked if they remain stable, i.e. their fluctuation is constant. Note that we monitor fluctuation, we do not check if the sample statistics exactly match the target statistics as would be done in an annealing setting.

- Multiple chains can be started and summary statistics over these multiple chains can be calculated. For example the average of the variograms of the multiple realizations generated by the multiple chains can be monitored.

In this paper, we rely on a simple visual check of the image to determine the number of iterations of the chain. Clearly, monitoring Markov chains for spatial stochastic simulation will be the subject of further research.

4.4 Local data exactitude

Many iterative sequential simulation methods (see e.g. Srivastava, 1992) honor the local data by freezing them at their locations on the simulation grid. Honouring local data requires two conditions

- At the data locations the local data must be exactly reproduced (exactitude property)

- Over a set of simulations the conditional variance, i.e. the variance of all simulated values at location \( \mathbf{u} \), should decrease continuously as location \( \mathbf{u} \) gets closer to a datum location. Otherwise discontinuities are created close to the data.
In the classical non-iterative sequential simulation methods these conditions are ensured because kriging ensures both exactitude and a continuous decrease of the conditional variances locally identified to the kriging variances. In iterative sequential simulation the first condition is met by freezing the data values, the second condition is less clear. To ensure this condition we alter the Metropolis-Hastings criterion (13) as follows

$$\alpha(s_{k^{\text{old}}} (u), s_{k^{\text{new}}} (u)) = \min \left\{ 1, \frac{\Pr \{ I(u; s_{k^{\text{new}}}) = 1 | s_i(u) \} p_{k^{\text{old}}} \} {\Pr \{ I(u; s_{k^{\text{old}}}) = 1 | s_i(u) \} p_{k^{\text{new}}} \} \times F \right\} \tag{14}$$

where we have added a second term to the Metropolis-Hastings criterion namely

$$F = \frac{\Pr \{ I(u; s_{k^{\text{new}}}) = 1 | (n) \} p_{k^{\text{old}}} \} {\Pr \{ I(u; s_{k^{\text{old}}}) = 1 | (n) \} p_{k^{\text{new}}} \} \tag{15}$$

where \((n)\) the set of \(n\) original sample data. The conditional probabilities \(\Pr \{ I(u; s_{k^{\text{new}}}) = 1 | (n) \}\) and \(\Pr \{ I(u; s_{k^{\text{old}}}) = 1 | (n) \}\) are determined using indicator kriging with the original sample data only, i.e.

$$\Pr \{ I(u; s_k) = 1 | (n) \} = p_k + \sum_{\beta=1}^{n} \lambda_i [i(u_\beta; s_k) - p_k],$$

for \(k = k^{\text{old}}, k^{\text{new}}\)

with \(i(u_\beta; s_k)\) denoting the indicators of the original sample data at the \(n\) data locations \(u_\beta\). The heuristic reasoning behind this new Metropolis-Hastings acceptance criterion is the following. As the simulated location \(u\) is closer to any data location the influence of the data should increase. However, the original sample data may not carry a lot of weight in the construction of the posterior probabilities \(\Pr \{ I(u; s_k) = 1 | s_i(u_j) \}\) using the trained neural network. For example if the template has 24 values, one local datum is but one single value among the 23 other values conditioning the posterior probability. In iterative simulation these 23 values carry little actual information when the simulation is started (initial values are random). Yet, the start of the simulation is critical because new structure emerges gradually through the updating process. To ensure that original sample data are weighted more, the second term (15) is added to the Metropolis-Hastings criterion. This second term becomes more important as the visited node \(u\) gets closer to the original sample data; at the data location the term B will only accept values that are equal to the data values. Far away from data we have

$$\frac{\Pr \{ I(u; s_{k^{\text{new}}}) = 1 | (n) \}} {p_{k^{\text{new}}}} \rightarrow 1$$

likewise,

$$\frac{\Pr \{ I(u; s_{k^{\text{old}}}) = 1 | (n) \}} {p_{k^{\text{old}}}} \rightarrow 1$$

Hence, relation (14) reduces to (13).
A more rigorous explanation of (14) is given in Appendix 2 and leads to further improvement the Metropolis-Hastings criterion:

$$\alpha(s_{\text{fold}}(u), s_{\text{new}}(u)) = \min\{1, A \times B\}$$  \hspace{1cm} (16)

with

$$A = \frac{\Pr\{I(u; s_{\text{new}}) = 1|s_t(u)\} p_{k\text{old}}}{\Pr\{I(u; s_{\text{old}}) = 1|s_t(u)\} p_{k\text{new}}}$$

$$B = \left(\frac{\Pr\{I(u; s_{\text{new}}) = 1|\{n\}\} p_{k\text{old}}}{\Pr\{I(u; s_{\text{old}}) = 1|\{n\}\} p_{k\text{new}}}\right)^{\omega(u)}$$

The power $\omega(u)$ is determined as follows

$$\omega(u) = 1 - \left(\frac{\sigma_{SK}(u)}{\sigma_{\text{max}}^{SK}}\right)$$  \hspace{1cm} (17)

The term $B$ in the Metropolis-Hastings criterion above has a power $\omega(u)$ for the following reason. As $u$ gets closer to an original sample value $u_\beta$, the term $B$ will become large if $s_{\text{new}}(u) = s(u_\beta)$ and will become small if $s_{\text{new}}(u) \neq s(u_\beta)$. In fact when $u = u_\beta$, then $\Pr\{I(u; s) = 1|\{n\}\} = 1$ for $s_{\text{new}} = s(u_\beta)$ and $\Pr\{I(u; s_{\text{old}}) = 1|\{n\}\} = 0$ for $s_{\text{old}} \neq s(u_\beta)$, i.e. $B$ becomes infinite. This means that term $B$ would dominate over term $A$ in the Metropolis-Hastings criterion, i.e. whatever $A$ is, the criterion is determined by the event $B$ only, thus close to data changes are determined by the kriging equations only. In order to overcome the dominant effect of kriging close to data locations, the term $B$ is downweighted using the power $0 < \omega(u) < 1$. This power should include some measure of distance between $u$ and the data, as the term $B$ should be more downweighted close to data. (17) proposes to use the kriging variance as a measure of distance in the term $B$. As $u$ comes closer to a datum, $\omega(u)$ tends to 1, hence exactitude is maintained through the use of kriging. As $u$ gets further away from $u_\beta$, then $\omega(u)$ tends towards 0, and the term $B$ becomes equal to 1, i.e. criterion (16) reduces to the original Metropolis-Hastings criterion (13).

### 4.5 Honoring soft data

#### 4.5.1 Metropolis-Hastings sampler with soft data

Conditioning should include also soft information. Soft data is usually defined on a larger support volume than that of hard data and gives only indirect information on the variable being simulated. Hence the information content of the soft data must be properly calibrated before being used.

To integrate soft data into geostatistical realizations using Markov chain sampling, the Metropolis-Hastings criterion must be changed. Consider a soft variable $Y(u)$ which
need not be categorical and is measured at certain locations constituting the soft data set \((n)_{\text{soft}}\).

To include that soft information in the Metropolis-Hastings sampler, the following result from Appendix 3 is used

\[
\frac{\Pr(A^*|B \cap C)}{\Pr(A|B \cap C)} = \frac{\Pr(A^*|B) \Pr(C|A^* \cap B)}{\Pr(A|B) \Pr(C|A \cap B)}
\]

where the following abbreviated notation has been introduced

\[A^* = \text{the new proposed class value } s_{\text{new}} \text{ at location } u\]
\[A = \text{the current class value } s_{\text{old}} \text{ at location } u\]
\[B = \text{the neighboring hard data class values } s_i(u) \text{ within a template around } u\]
\[C = \text{the soft information } y(u) \text{ at location } u\]

Here \(C\) is collocated information only, although the method can be extended to non-collocated information.

The Metropolis-Hastings acceptance criterion with soft data is then defined as

\[
\alpha(s_{\text{old}}(u), s_{\text{new}}(u)) = \min\left\{1, \frac{\Pr(A^*|B) \Pr(C|A^* \cap B) Q(A^*, A)}{\Pr(A|B) \Pr(C|A \cap B) Q(A, A^*)}\right\}
\]

or in terms of spatial variables

\[
\alpha(s_{\text{old}}(u), s_{\text{new}}(u)) = \min\left\{1, \frac{\Pr(I(u; s_{\text{new}}) = 1|s_i(u)) \Pr(Y(u) = y(u)|s_{\text{new}}(u), s_i(u)) p_{\text{old}}}{\Pr(I(u; s_{\text{old}}) = 1|s_i(u)) \Pr(Y(u) = y(u)|s_{\text{old}}(u), s_i(u)) p_{\text{new}}}\right\}
\] (18)

The Metropolis-Hastings criterion has now three distinct terms. One term relates to the ratio of the proposal matrix \(Q\) as before. The second term \(\Pr(A^*|B)/\Pr(A|B)\) relates to the primary data only and is the original Metropolis-Hastings criterion (13) without considering any soft information. The third term \(\Pr(C|A^* \cap B)/\Pr(C|A \cap B)\) relates to a ratio of probabilities of obtaining a soft information \(C\) given the current hard information \(A \cap B\) and \(A^* \cap B\). \(\Pr(C|A \cap B)\) is termed the likelihood of obtaining the soft information.

The likelihood \(\Pr(C|A \cap B)\) depends on the existing relation between \(C\), the soft data, and \(A \cap B\), the collocated and neighboring hard information at that particular stage in the Markov chain simulation. That relation is termed a forward model, represented by a (non-linear) function \(g\)

\[
C = g(A \cap B)
\] (19)
The forward model represents the physics (seismic/flow) governing the soft information. \( g \) is usually highly non-linear and approximated in simulation models (flow simulation/seismic ray tracing). Hence if the physics were perfectly known then given \( A \cap B \), \( C \) would also be perfectly known and \( \Pr\{C|A \cap B\} \) would be a spike. This ideal situation never occurs in reality for two reasons:

- **Measurement errors**: The soft information is not measured exactly due to measurement errors, it is denoted as \( \hat{C} \). Then \( \Pr\{C|A \cap B\} \) determines the probability distribution, of the underlying true, error free measurement, \( C \) given \( A \cap B \). The expected value of those probability distribution usually taken to be the measurement \( \hat{C} \).

- **Model errors**: Two types of model errors are distinguished
  
  - The function \( g \) is not known exactly but can only be approximated in, for example, a simulation model using finite element/ finite difference methods. \( \Pr\{C|A \cap B\} \) models the approximations made due to incomplete knowledge about the true forward model (19).
  
  - The hard information \( A \cap B \) is incomplete. Ideally, \( A \cap B \) should represent the exhaustive field of hard information, e.g. facies values at each location are used in order to determine the seismic at location \( u \) with the forward model (19). For practical purposes, \( A \cap B \) represents only a limited neighborhood around location \( u \) where the soft data was acquired. Hence \( \Pr\{C|A \cap B\} \) models the errors made due to this assumption.

In the next section, neural networks are used to approximate the forward model \( g \) and \( A \cap B \) are taken to be a limited neighborhood. The advantage of neural networks is that, once trained, they can be evaluated CPU-effective. Such trained neural networks are fast proxies for the possibly CPU-expensive evaluation of the forward model (19). We have to take into account the errors made by these approximations and hence, determine what \( \Pr\{C|A \cap B\} \) is.

### 4.5.2 Calibration of soft data using neural networks

Training images can be used to directly determine the conditional probabilities

\[
\Pr\{C|A \cap B\} = \Pr\{Y(u) = y(u)|s_k(u), s_l(u)\}
\]

To obtain such conditional probabilities, a dataset has to be constructed. Given a training image in the variable \( S(u) \), we can forward simulate a physical process (seismic, flow) to obtain a corresponding training image in the soft data variable \( Y(u) \) (see Figure 1). Both
training images are scanned (Figure 4a) and at each location \( \mathbf{u} \) of the hard data training image a pixel class value and its neighbors (defined within a template) are collected and on the corresponding soft data training image at the same location the soft information \( y(\mathbf{u}) \) is collected. We limit ourselves to collocated soft information. This provides a dataset
\[
\{s(\mathbf{u}_j), s_t(\mathbf{u}_j); y(\mathbf{u}_j)\}, j = 1, \ldots, N
\]
(20)
\( N \) is the total amount of data vectors that can be obtained by scanning the hard and soft data training image. Using this dataset, neural networks can be used to determine the conditional probabilities \( \Pr\{Y(\mathbf{u}) = y(\mathbf{u})|s_k(\mathbf{u}), s_t(\mathbf{u})\} \). Three methods are proposed (see Figure 4 b,c,d)).

- **method 1**: the inputs of the neural network are the hard information \( \{s(\mathbf{u}); s(\mathbf{u} + \mathbf{h}_\alpha), \alpha = 1, \ldots, n_t\} \), the target output of the network is the soft information \( y(\mathbf{u}) \) (Figure 4b). The network is trained using the sum-of-squares error criterion
\[
Ef_{soft} = \frac{1}{2N} \sum_{k=1}^{K} \sum_{j=1}^{N} (y^o(s(\mathbf{u}_j), s_t(\mathbf{u}_j); \theta) - y(\mathbf{u}_j))^2
\]
y\(^o\) the output of the neural network that should match the target output provided by the training set. As the output of a trained network can be interpreted as a conditional expectation
\[
y^o(s(\mathbf{u}_j), s_t(\mathbf{u}); \theta) = \mathbb{E}[Y(\mathbf{u})|s(\mathbf{u}), s_t(\mathbf{u})]
\]
the result will be the conditional expectation of the soft data given the hard information. Using a Gaussian distribution with as mean this conditional expectation and as variance a fixed value \( \sigma \) we obtain the conditional distribution as
\[
\Pr\{Y(\mathbf{u})|s(\mathbf{u}), s_t(\mathbf{u})\} \sim \mathcal{N}(\mathbb{E}[Y(\mathbf{u})|s(\mathbf{u}), s_t(\mathbf{u})], \sigma)
\]
How the variance \( \sigma \) is determined in practice is illustrated on a case study in the next section.

- **method 2**: the neural network can be used not only to find the conditional expectation but also the conditional variance in the following way, see Figure 4c. First a network is trained as proposed above. The output of this network represents as before the conditional expectations of the output variables \( Y(\mathbf{u}) \) given the input variables. For each input datum vector \( \{s(\mathbf{u}); s_t(\mathbf{u})\} \) an output prediction can be calculated using the trained weights of the network. Each of these calculated outputs is a conditional expectation. That conditional expectation is then substracted from the target output values \( y(\mathbf{u}) \), the result is squared and used as target outputs for a second neural network, which has the same input variables \( \{s(\mathbf{u}); s_t(\mathbf{u})\} \). The outputs of the second network then represent the conditional expectation of
\[
(Y(\mathbf{u}) - \mathbb{E}[Y(\mathbf{u})|s(\mathbf{u}), s_t(\mathbf{u})])^2
\]

24
and thus are conditional variances
\[ \text{Var}[Y(u)|s(u), s_t(u)] = E[(Y(u) - E[Y(u)|s(u), s_t(u)])^2] \]

The conditional distribution now becomes
\[ \text{Pr}\{Y(u)|s(u), s_t(u)\} \sim N(E[Y(u)|s(u), s_t(u)], \text{Var}[Y(u)|s(u), s_t(u)]) \]

- **method 3**: The neural network output can itself be the conditional distribution, see Figure 4d. This methodology is developed in Caers (1998) using a mixture of basic distribution as a model for the conditional distribution and can be applied to find the conditional distribution of the soft information.

These three methods have an increased complexity in the use of neural networks. Method 1 predicts the soft data, given the hard information. In method 1 a fixed variance \( \sigma \) (homoscedasticity) is used to model the uncertainty of that prediction. Method 2 uses two networks instead of one, the second network is used to determine the conditional variance of the soft data given the hard information. Method 2 is an improvement over method 1 because in the former it is not assumed that there is a constant uncertainty (in method 1 provided by \( \sigma \)) that is the same for all possible hard information. In method 2, the conditional variance depends on the predicted value (heteroscedasticity). Method 3 goes further and does not assume any Gaussian distribution as in method 1 and 2, the distribution is in fact a mixture of Gaussian distributions (Caers, 1998).

### 4.6 Honoring global constraints

Various global constraints should be imposed on the resulting realizations. Such constraints could be e.g. histograms, variograms, vertical and horizontal proportion curves.

The Metropolis-Hastings sampler leaves the flexibility to explicitly honor such global constraints. To honor a specific constraint such as a histogram one starts with an initial image that already has the constraint matched, e.g. has the correct (global or vertical) proportion of facies. Then an objective function is defined that measures the difference between the target histogram and the histogram at every iteration step in the Markov chain. The objective function is used to ensure that the simulation does not wander too far away from the initial correct proportion. As known, the Metropolis-Hastings sampler has two parts: a proposal phase defined by a proposal matrix \( Q \), where a new value is proposed and an acceptance criterion \( \alpha \). The key idea is to use the proposal part, to propose new values that do not violate *too much* the target global constraints, allowing for some fluctuations around the target value. The result is still a Metropolis-Hastings sampler, because we simply used the added freedom provided in the matrix \( Q \) used in the proposal part.

25
The objective function is a sum-of-square errors measuring the distance between the target and what is currently simulated. In the case of the histogram the objective function measures the sum-of-squares error between the simulated and target quantiles \( q_m, m = 1, \ldots, M \)

\[
O(s^{(l)}) = \sum_{m=1}^{M} (q_m - q_m(s^{(l)}))^2
\]

where \( q_m(s^{(l)}) \) are quantiles of the global histogram at the current simulation step \( l \) and \( s^{(l)} \) are the current simulated values at each grid node in the iterative simulation. In the next step \((l + 1)\) of the iterative simulation one single value is changed. If the objective function is easily updatable, i.e. if we can calculate the new objective function \( O(s^{(l+1)}) \) from the current one easily as a function of the newly proposed class value \( s_{k{\text{new}}} \), the difference between the old and the new objective function can be expressed as a function of the current and new class value proposed at location \( u_j \)

\[
O(s^{(l)}) - O(s^{(l+1)}) = f(s_{k{\text{new}}}(u_j), s_{k{\text{old}}}(u_j))
\]

A decrease of the objective function is favorable, since it leads to a better match of the global statistics. Such decrease is accepted all the time, while an increase is accepted with certain probability, given by an exponential distribution. Therefore, the objective function enters into the Metropolis-Hastings sampler through the \( Q \) matrix as follows

\[
Q(s^{(l)}, s^{(l+1)}) = C \exp(-O(s^{(l+1)})/C)
\]

hence the ratio of \( Q(s^{(l)}, s^{(l+1)}) \) over \( Q(s^{(l+1)}, s^{(l)}) \) becomes

\[
R = \frac{Q(s^{(l)}, s^{(l+1)})}{Q(s^{(l+1)}, s^{(l)})} = \exp(-(O(s^{(l+1)}) - O(s^{(l)}))/C)
\]

(21)

The constant \( C \) determines how much the user would like to honor that constraint. If \( C \) is low the histogram or vertical proportion will be exactly met, if \( C \) is higher one accepts some fluctuations of the simulated histogram or proportions around the pre-set target. Contrary to annealing (Deutsch, 1992) this parameter is not a tuning parameter and there is no temperature to change. Usually the initial random image that is constructed prior to the iterative simulation is such that it already has the target histogram and proportion. The introduction of the objective function is simply to maintain the correct histogram and proportions in the subsequent iterations. This ratio \( R \) enters the Metropolis-Hastings acceptance criterion (13), (14) or (16). For example

\[
\alpha(s_{k{\text{old}}}(u), s_{k{\text{new}}}(u)) = \min \left\{ 1, \frac{\Pr\{I(u; s_{k{\text{new}}}) = 1|s_{k}(u)\}}{\Pr\{I(u; s_{k{\text{old}}}) = 1|s_{k}(u)\}} \times R \right\}
\]

Note that any objective function must be easily updatable after each iteration. This is the case when the constraint is a proportion curve, a variogram or a histogram.
5 Case study

5.1 The data set

To demonstrate the concepts of a pattern recognition-based geostatistics, the following synthetic case study is designed. Figure 5 shows a horizontal section of a fluvial channel-type reservoir of size $100 \times 150$ pixels, with two facies present: channel-sand facies and mudstone facies. This section is considered as the true reservoir. The variable under study is thus of binary facies-type with two possible classes: $s_1 = 1$ channel, $s_2 = 0$ mud. The channels are simulated using an object-based technique fluvsim (Deutsch and Wang, 1996). The sand proportion is 50%. The conditioning data consists of two horizontal wells, see Figure 5. Each well has 37 equally spaced observations of facies type, totalling 74 conditioning data. Seismic data is provided also in Figure 5 to serve as soft data. The seismic data is obtained by a linear averaging process mimicking the resolution of an actual seismic survey as will be explained later.

For training and calibration, a training image is provided in Figure 6. The training image shows patterns of channeling similar to the true reservoir. Using the same linear averaging process as for the true reservoir a seismic campaign is forward generated on the training image and shown in Figure 8. In practice, one would expect that the forward simulation process of the seismic on the training image would yield different structures than the measured seismic on the true reservoir, because the true physics of wave propagation is only approximated in the forward simulation process. Also the training image may have a geological structure different from that of the true reservoir.

The intention is to perform first unconditional simulation, next conditional simulation with hard data only, finally conditional simulation with hard and soft information. In a real case study these represent actual steps in the construction of reservoir models using pattern recognition techniques. Unconditional simulation is the ultimate test to check structure reproduction, without any help from conditioning data: one checks if the neural network has properly learned the texture of the training image and if the Metropolis-Hastings sampler has succeeded to reproduce this texture in the iterative simulation.

5.2 Unconditional simulation

Consider first unconditional simulation. Due to the long-range structure of the channels a multi-grid approach is adopted (Tran, 1994). First simulation is performed on a coarse grid of $33 \times 50$ pixels. The $100 \times 150$ training image is scanned with the star shaped template shown in Figure 7a. The distance between the pixels in the star is 3 map units for the coarse grid. The dataset comprises of the set of vectors

$$t(u_j) = \{s(u_j); s(u_j + h_\alpha)\}, \quad \alpha = 1, \ldots, 12, \quad j = 1, \ldots, 14016$$
From this dataset, we randomly extract 50% of the vectors as training and 50% for validation. A single hidden-layer feed-forward neural network with 30 hidden layers is trained to determine the posterior probability of obtaining a channel sand in the central location of the template given the facies data in the neighboring template locations. The training error calculated using Eq. (6) and validation error calculated using Eq. (11) are shown in Figure 8. Training is stopped after 18 iterations when the validation reaches its minimal value. Figure 8 shows the importance of using a validation set: if the simulation is not stopped after 18 iterations, the training error further decreases and the neural network will overfit the training patterns. Next simulation is performed on two successively finer grids, conditioned each time to the simulated values generated on the coarser grid. The grids are 50 × 75 pixels and 100 × 150 pixels respectively. For each of these grids a different neural network must be trained.

Using the trained neural networks, four unconditional simulations are shown in Figure 9. Each unconditional simulation is constrained to a global sand proportion of 50% by introducing the equation (21) in the Metropolis-Hastings sampler. The simulation reproduces well the channeling of the training image, the channel thickness is well reproduced, the channels are however somewhat less undulating than in the training image. Figure 9 shows that the 25 simulated proportions match well the target proportion of 50% sand.

5.3 Conditional simulation

Conditional simulations are constructed in the same fashion as unconditional simulations, i.e. with the same templates and the same neural networks, but they are constrained to the data along the two wells. 25 realizations are constructed using the Metropolis-Hastings criterion

$$\alpha(s_{k\text{old}}(u), s_{k\text{new}}(u)) = \min \{1, A \times B\}$$

with

$$A = \frac{\Pr\{I(u; s_{k\text{new}}) = 1|s_{i}(u)\} \Pr\{I(u; s_{k\text{old}}) = 1\}}{\Pr\{I(u; s_{k\text{old}}) = 1|s_{i}(u)\} \Pr\{I(u; s_{k\text{new}}) = 1\}}$$

$$B = \left( \frac{\Pr\{I(u; s_{k\text{new}}) = 1|n\} \Pr\{I(u; s_{k\text{old}}) = 1\}}{\Pr\{I(u; s_{k\text{old}}) = 1|n\} \Pr\{I(u; s_{k\text{new}}) = 1\}} \right) \omega(u)$$

The power $\omega(u)$ are obtained from the kriging variance as follows:

$$\omega(u) = 1 - \left( \frac{\sigma_{SK}(u)}{\sigma_{SK}^{\max}} \right)^{\omega}$$

The kriging variance $\sigma_{SK}(u)$ is obtained from a simple kriging using the 74 well data.

Consider first the case $\omega = 0$, i.e. the term B becomes one and it cancels out of the Metropolis-Hastings criterion. Figure 10 shows four conditional simulations, each
conditioned exactly to the same well data. The simulations look in structure similar to the unconditional simulation, the channel location is better controlled due to the presence of conditioning data. Figure 10 also shows the histogram of the sand proportion of 25 realizations which matches well the target proportion of 50%. Figure 11 shows the E-type estimate and the conditional variance of these 25 realizations. The E-type estimates are conditional probabilities (conditional to the well data) for being in sand. Both E-type and conditional variance show that the channeling is well conained close to the wells and less further away from the wells.

In some cases, angle data providing channel orientation are available from well-logging. Kriging with spatially changing anisotropy (Soares, 1992) is implemented to account for the change in channel direction in the following way. The angles of changing anisotropy enter in the covariances of the right-hand side of the kriging equations only (see Soares, 1992 for details). The term $B$ in the Metropolis-Hastings sampler is now included and we take $\omega = 0.2$. The kriging with spatially changing anisotropy is used to determine the probabilities

$$Pr\{I(u; s_{knew}) = 1|(n)\} \quad \text{and} \quad Pr\{I(u; s_{kold}) = 1|(n)\}$$

Hence prior to simulation, a kriging map of the probability of any location $u$ being sand, given the well and angle data is constructed. The angle data used to construct the kriging map are shown in Figure 12, next to the kriging map. The kriging result can be read from that map and the term $B$ constructed at each node of the simulation grid. Close to the well, due to the inclusion of kriging in the term $B$ in the Metropolis-Hastings sampler, the directionality of the channels will be reproduced. Figure 13 shows four simulations conditioned to the angle data. The E-type estimate of 25 simulation is shown in Figure 14. Comparing Figure 14 to Figure 11 shows that the angle data has an effect on the directionality of the channels, near the well. The uncertainty – as is reflected by the conditional variance in Figure 14 – of the location of channels has decreased near the wells and that decrease is highest in the direction of the angle information (compare Figure 11 and 14).

### 5.4 Conditional simulation with soft data

To mimick a forward simulation of seismic data, an easier linear averaging process was performed. In an actual study, one would use ray tracing to include the physics of wave propagation.

The seismic datum $y(u)$ at any location $u$ is constructed as a linear average of the neighboring facies values

$$y(u) = \frac{1}{25} \sum_{a=1}^{25} 2(s(u + h_a) + 2)$$

29
where the $\mathbf{h}_a$'s define the template shown in Figure 7b. The seismic datum $y(\mathbf{u})$ is thus not only related to the collocated facies-type information at $\mathbf{u}$ ($\mathbf{h}_a = 0$) but also neighboring facies-types. Finally, the seismic data is linearly scaled to be in the $[0, 1]$ interval.

The idea of generating seismic using this particular linear average is to obtain a seismic image that reflects certain properties of a true seismic processing. Seismic data informs a larger volume than hard data measured at wells, hence the seismic data has a much smoother behavior than the real underlying petrophysical property or facies type. Seismic generally does not reveal small individual channels but may reveal a zone with high concentration of channel sands. Furthermore, seismic data might be prone to various seismic processing artifacts. Figure 5 and 6 show that both features (smoothness and artifacts) are present in the constructed seismic data on both true and training reservoir. The seismic data tells roughly where channels are but does not reveal any individual channel. In some cases the seismic may even create ”phantom channels”, which look like channels but are artifacts. The seismic data needs to be properly calibrated to account for these effects.

**Calibration of the soft data**

In order to use the Metropolis-Hastings sampling criterion (18) for integrating soft data, the conditional distribution of a seismic measurement at location $\mathbf{u}$ given any collocated and neighboring facies-types must be established. To determine this conditional probability, method 1 of section 4.5.2. is used, i.e. the conditional expectation is determined from a calibration between the hard-data training image and the soft data training image. A neural network is used to obtain this calibration and we will term this neural network the calibration neural network.

To obtain the conditional expectation of the seismic (i.e. the likelihood of the seismic), a training and validation set must be available. The inputs of the calibration neural network are

- The facies type at the seismic data location $\mathbf{u}$.
- The facies at neighboring locations with respect to $\mathbf{u}$, given by the template in Figure 7c.

The output of the calibration neural network is then the seismic $y(\mathbf{u})$ event recorded at $\mathbf{u}$. The set of training and validation data are then

$$\{s(\mathbf{u}); s(\mathbf{u} + \mathbf{h}_a), \alpha = 1, \ldots, 12; y(\mathbf{u})\}, j = 1, \ldots, N$$

(22)

with $\mathbf{h}_a$ defined by the template. Again, a 50/50 division of the data is made to construct a validation and training set. Note that the template used for calibration (Figure 7c) is
different from the template used for constructing the seismic (Figure 7b). Indeed, were the
same template used, the neural network might be able to exactly reproduce the forward
seismic simulation which is not the purpose of the exercise. This mimicks a realistic
situation because in general the neural network will rarely be able to reproduce exactly
the forward seismic simulation. Figure 15 shows the training and validation error when
training a neural network with 30 hidden layers to the dataset (22). The validation error
(Figure 15) reaches a minimum after 33 iterations.

After the network parameters are determined, the performance of the training must be
evaluated. For that purpose, the data input vectors used for training are fed to the trained
calibration neural network which calculates the corresponding output for each of these
input vector. The calculated outputs are the network predictions. The accuracy of these
prediction can be calculated by comparing the predictions with the target output seismic
that corresponds to the training data input vector used for calculating that prediction.
This neural net output is compared to the target data \( y(\mathbf{u}) \) on a scatterplot in Figure 16a.
The scatterplot shows the difference between what the neural network has learned from
the input and what it should have learned, i.e. the target output data. The top of Figure
16a shows the accuracy of prediction for the training set, the bottom for the validation set.
The correlation coefficients are 0.86 and 0.75 for training and validation respectively. The
calibration neural network does not perform a perfect job in predicting the seismic given
the facies types at neighboring locations. In other words, with this trained calibration
neural network, given one knows at location \( \mathbf{u} \) the set of collocated and neighboring facies
types, one cannot exactly predict what the seismic will be at \( \mathbf{u} \). Hence in the simulation
phase, when using the Metropolis-Hastings sampler, the seismic will not be matched
exactly, because the neural network cannot exactly predict the seismic. A modelling error
needs to be included, due to the imperfect modelling of the seismic using the neural nets.
Therefore, in method 1 of section 4.5.2. a parameter \( \sigma \) is introduced that reflects the
uncertainty due to that modelling error. Figure 16b shows how that modelling error is
obtained: for a given predicted seismic, a histogram of the "true" seismic is constructed
using the validation data. The variance of that histogram is calculated and provides the
value for that \( \sigma \). We take the approximation that the variance \( \sigma^2 \) is the same for any
predicted seismic (homoscedasticity) and that the distribution model for the histogram
is Gaussian with as expectation the prediction itself and as standard deviation \( \sigma \). Here,
\( \sigma = 0.2 \) for all predictions.

Simulation

Four conditional simulations with the soft data are shown in Figure 17. The realizations
are also conditioned to the hard data (no angle information). The channel thickness has
slightly increased and the channels are slightly less continuous, but the channel locations
are now much more restricted. To check how well the soft data is honored, a forward simu-
lation of the seismic is run on the four simulations, i.e. the same linear averaging process
that produces the seismic training data is applied. Figure 18 shows the four resulting forward simulations of the seismic. Clearly, the soft data is not exactly reproduced, as expected from the inclusion of a modelling error. To check better the reproduction of the soft data, the difference between the seismic data and the forward simulated seismic is calculated at each location of the grid, for each conditional simulation. The difference at each location is then averaged over the 25 simulated realizations. Figure 19 shows a plot of the average difference between the seismic data and the simulated seismic and a histogram of these average differences. The global average difference is exactly 0 (Figure 19 (middle)). The plot shows that there is still significant autocorrelation existing in this average difference image. This suggests that the use of a homoscedastic variance $\sigma^2 = 0.2$ and the Gaussian distribution is not the most optimal method. It suggests that the accuracy of prediction of the seismic data using the calibration neural network depends on the value of the seismic datum itself, hence the accuracy is not the same everywhere on the grid, i.e. the parameter $\sigma$ should be made location-dependent (heteroscedastic). Therefore, either method 2 or 3 of section 4.5.2. should be preferred. This is definitely an area of future research.

To further check how well the soft data is reproduced in the simulations, the correlation coefficient of the forward simulated seismic and the true seismic data is calculated for 25 realizations. The histogram of these 25 correlation coefficients is shown in Figure 19(bottom). The correlation ranges between 0.6 and 0.7 which is lower than the correlation of the validation data set (0.75, see Figure 16a). The inclusion of soft data has a clear impact on the uncertainty space of the 25 realizations. The E-type estimate and conditional variance are shown in Figure 20 and shows a clear reduction of uncertainty with respect to the simulation with only hard data (Figure 11). This is particularly clear when comparing the conditional variances. The channeling is now restricted to a narrower uncertainty range.

5.5 Conditional simulations constrained to proportions

Often proportion curves of sand-to-gross ratios are available and should be included in the reservoir models. Proportion curves can be obtained from well-log information, seismic data or from an interpretation of the depositional environment (coarsening upward, fining upward). Figure 21 shows a target proportion curve in the North direction, i.e. in the same direction as the wells are alligned. This proportion curve is obtained from calculating the true proportion from the true reservoir and by adding some noise. Figure 21 shows that there is a periodicity in the proportion curve with three peaks reflecting a higher proportion of channels. The proportion curve is constructed such that the global proportion of sand in the reservoir is still 50%. Matching the proportion curve thus entails also a match in a global channel-sand proportion of 50%.

Four conditional simulations (to the same well data as above) are presented in Figure
22 that are constrained to the proportion curve. The proportion curves of 25 simulated realizations versus the target proportion curves are given in Figure 23. The simulated proportions match the target proportion with some fluctuation while maintaining the channeling within the reservoir. The impact of the vertical proportion on the channeling can be studies by plotting the E-type estimate and conditional variance of 25 realizations (Figure 24) and comparing this result with the E-type and variance calculated on realizations without the proportion curve given in Figure 11. Figure 24 shows that there is difference in E-type and variance compared to Figure 11: the mean and variance reflect that the channeling is more restricted with the proportion curve.
6 Conclusions and future research

The goal of any geostatistical reservoir characterization is to construct reservoir models that are *maximally data charged*. The data available for such characterization covers a very broad spectrum of scale and accuracy.

First, there is the geological interpretation which depicts the style of the underlying heterogeneity structure. This believed structure is termed the prior structural model of the reservoir: such model represents information about types of facies, facies associations, distributions of petrophysical properties within the facies, fracture models, etc. The prior structural model is not conditioned to any local information. A realization of such prior model is termed in geostatistics an unconditional simulation. Secondly, conditioning local information is available and is used to constrain (anchor) the prior model. Such information includes well data of facies and petrophysical variables, dipmeter information, fracture orientation at wells, seismic well logs, cross-tomography, 3D and 4D seismic surveys, well test and production data.

Prior models

The current geostatistical methodology relies on the variogram to represent the prior structural model, either in a multivariate Gaussian framework (sgsim) or in a non-Gaussian framework (isim). However the variogram, as a two-point statistics, is too limited to capture the complexity of the underlying geology. Multiple-point statistics must be called for. If geological models are available as training images, depicting a complex geological pattern which is beyond the modelling capability of the variogram, then that information should be included in the reservoir models. In this paper it is shown that statistical pattern recognition can be used to extract complex patterns from training images and reproduce them in the actual reservoir models. The difficulty of choosing a prior structural models has now been mapped into to the difficulty of drawing a training image. The methodology relies on the use of neural networks to model local conditional distributions by regarding all local information at the same time, instead of the classical two-by-two information used in kriging. However, the use of such pattern recognition techniques raises new issues

- The issue of stationarity and exportability now becomes absolutely crucial. The goal is not to use the training image as a pseudo wall-paper design, but to recognize and decide which pattern is important in the training image and could be exported. Statistical pattern recognition captures these patterns in probability terms. It is not yet clear what essential patterns are actually captured by these probability distributions. The solution of such issue might be found in the "randomization" of the template geometry used to scan the image. In the current methodology, the
template geometry is fixed, chosen by the user. Some template geometries might capture well the pattern present in the training image while others do not. It is therefore important to find a methodology to determine the "optimal" template geometry, where optimal includes the ability of a template to yield back the structure of the training image in the unconditional simulations. In fact, one can rely on one important paradigm: if a particular pattern is essential in the training image it will repeat itself frequently over the entire training image. Noisy patterns are usually uncorrelated to each other. Statistical image analysis methods such as unsupervised clustering and feature extraction might prove useful to extract such important patterns, by filtering the noisy patterns. Unsupervised clustering amounts to finding correlated repetitive patterns in the image and grouping them.

- Multiple training images can be available, each depicting important differences in interpretation of the underlying geological. Each such image thus depicts a different prior structural model. In such case, one could use different neural networks on each of the training images. The outputs of such network could then be linearly weighted into a single output, the weights reflecting ones prior belief (one training image is deemed more likely than an other) about each single training image.

- It might be difficult to construct 3D training images, but multiple two dimensional cross-section (training cross-sections) in various planes might be available, e.g. from outcrop modelling. The neural network methodology needs to be extended to be able to include the structural information from these various cross-sections. Alternatively, CAD can be used to combine these 2D cross-sections into 3D training images.

**Data calibration**

Prior structural models must be conditioned to various types of local data. For conditioning to so-called soft information, the relationship between soft and hard variable (facies type, petrophysical property) must be established. This is the calibration of soft information. The current geostatistical methodology relies on two-point models for such calibration, such as correlation coefficients or cross-variograms or single point averages (well test), or on a post-processing of the reservoir models, for example to match flow-history.

The soft information (seismic/flow) has typically a complex, non-linear, multi-point relation with the hard variable. For example a well test response has a non-linear relation with the multiple permeability value adjacent to the well. That relation is multiple-point, i.e. it relates to all permeability taken jointly, it is not some simplified single-point average. For a discussion on well test calibration using multiple point, we refer to Srinivasan (1999). Neural networks can serve as a methodology that captures the multiple-point relation.
between soft data and hard variables. The training image is used to construct a soft-data training image by forward simulation of the physical process (seismic/flow) governing the soft data. The pair of hard-data training image and the soft-data training image can then be used to calibrate the relation between soft and hard. That relation serves as a CPU-effective proxy for the full CPU-expensive forward simulation. Note that the relation is still a statistical relation, i.e. the neural network can only predict the soft data from the hard variables in a statistical sense.

The calibrated soft data must be reproduced into the reservoir models. The goal is now to construct reservoir models such that if a forward simulation of the physical model is run on the reservoir model, the soft data would be reproduced. That reproduction need not be exact: data errors and model errors might exists. Data errors reflect the inaccuracy of the data measurement itself, while model errors reflect the inaccuracy of the proxy in approximating the full forward model.

**Simulation**

In this paper, the reproduction of a prior model and the conditioning of information is performed using the Metropolis-Hastings sampler. The Metropolis-Hastings sampler is an iterative technique for constructing stochastic reservoir models as opposed to most current geostatistical techniques which are non-iterative. The iterative techniques allow to construct a much larger class of reservoir models because they are much more flexible as shown in this paper, yet they raise an additional crucially important issue: convergence. The congenial property of the Metropolis-Hastings sampler is that it converges, and moreover it converges independently from what one starts or how one iterates, a property not shared by the popular simulated annealing methods. However, the number of steps after which convergence is reached is not known, an area of considerable future research.

As opposed to simulated annealing, the Metropolis-Hastings sampler does not have an objective function that could contain conflicting objectives hampering convergence. In fact the Metropolis-Hastings has only one single objective: draw samples from an implicit or explicit random function. In our case the random function is always implicit, which means that only certain properties of the random function are known. Such properties are the prior structural model and the conditioning data (soft and hard). The prior structural model and the way the conditioning is performed are consistent because the same training image is used to extract both the prior model and to calibrate the soft information. However that prior structural model need not be consistent with the actual subsurface geology. The impact of such inconsistency (wrong prior) must be further researched.

The combination of Metropolis-Hastings sampler with neural networks for data cali-
bration and pattern recognition provides a full forward modelling perspective to stochastic reservoir characterization. In this paper, we suggest how the traditional CPU-intensive inverse modelling techniques for data integration could be replaced by a more CPU-effective forward modelling approach.
7 References


SRINIVASAN, S., 1999. This volume.


STREBELLE, S., 1999. This volume.


Appendix 1

Sum-of-squares error function

In the main text, it is shown that neural networks are trained with a sum of squares error function using indicator information (provided by the training image) as target output data for the neural network. We interpreted the output of the neural network as the expectation of the indicator of the central value in the template given the neighboring classes in that template. The neighboring class values are the inputs for the network. As the expectation of an indicator is a probability (a key paradigm in geostatistics), the output of the neural network can be interpreted as a conditional probability. The conditioning is done on the input of the network.

In this appendix, it is shown that the output of a neural network trained with a sum-of-squares criterion can indeed be interpreted as an expectation and hence it is a conditional probability:

\[ E[I(u; s_k)|s_i(u)] = \text{Pr}\{I(u; s_k) = 1|s_i(u)\} \]

The sum of squares error function can be calculated under a limit of a large data set

\[
Ef = \lim_{N \to \infty} \frac{1}{2N} \sum_{k=1}^{K} \sum_{j=1}^{N} (y_k(s_i(u_j), \theta) - i(u_j; s_k))^2
\]

\[ = \frac{1}{2} \sum_{k=1}^{K} E[(y_k(S_t(u_j), \theta) - I(u_j; s_k))^2] \]  \hspace{1cm} (23)

To find the minimum of the function \( Ef \) one proceeds as follows. The expected value calculated in (23) is calculated over categorical variables \( I \) and \( S \). The expectation of a categorical variable is expressed as a sum

\[
E[(y_k(S_t(u_j), \theta) - I(u_j; s_k))^2] = \sum_{all} \sum_{i(u_j; s_k) = 0}^1 (y_k(s_i(u_j), \theta) - i(u_j; s_k))^2 p(s_i(u_j), i(u_j; s_k))
\]

\[ \text{where } p(s_i(u_j), i(u_j; s_k)) \text{ is the joint multivariate discrete distribution of the variables } S(u_j) \text{ and } I(u_j; s_k). \]

This joint multivariate distribution can be decomposed into a conditional and marginal

\[ p(s_i(u_j), i(u_j; s_k)) = p(i(u_j; s_k)|s_i(u_j)) p(s_i(u_j)) \]  \hspace{1cm} (25)

Also the term \( (y_k(s_i(u_j), \theta) - i(u_j; s_k))^2 \) can be expanded as

\[
(y_k(s_i(u_j), \theta) - i(u_j; s_k))^2 = (y_k(s_i(u_j), \theta) - E[I(u_j; s_k)|s_i(u_j)])^2
\]

\[ + (y_k(s_i(u_j), \theta) - E[I(u_j; s_k)|s_i(u_j)]) (E[I(u_j; s_k)|s_i(u_j)] - I(u_j; s_k))
\]

\[ + (E[I(u_j; s_k)|s_i(u_j)] - I(u_j; s_k))^2 \]
The second term in this expression combined with expression (24) and (25) becomes

$$\sum_{a l l \, n_t} \sum_{i(u_j; s_k) = 0}^1 (y_k(s_t(u_j), \theta) - E[I(u_j; s_k)|s_t(u_j)])$$

$$(E[I(u_j; s_k)|s_t(u_j)] - i(u_j; s_k)) p(s_t(u_j), i(u_j; s_k))$$

$$= \sum_{a l l \, n_t} \sum_{i(u_j; s_k) = 0}^1 (y_k(s_t(u_j), \theta) - E[I(u_j; s_k)|s_t(u_j)]) p(s_t(u_j), i(u_j; s_k))$$

$$\sum_{i(u_j; s_k) = 0}^1 (E[I(u_j; s_k)|s_t(u_j)] - i(u_j; s_k)) p(i(u_j; s_k)|s_t(u_j)) = 0$$

The first and third term lead to the following expression for the error function

$$E f = \frac{1}{2} \sum_{k=1}^K \sum_{a l l \, n_t} (y_k(s_t(u_j), \theta) - E[I(u_j; s_k)|s_t(u_j)])^2 p(s_t(u_j))$$

$$+ \frac{1}{2} \sum_{k=1}^K \sum_{a l l \, n_t} (E[I(u_j; s_k)|s_t(u_j)] - i(u_j; s_k))^2 p(s_t(u_j))$$

The last term in this expression does not depend on the weights $\theta$, hence can be ignored for the minimization of the sum of squares error. The first term is non-negative, the minimum of the error function occurs when this term is zero, which occurs when a set of weights $\theta$ can be found for which

$$y_k(s_t(u_j), \hat{\theta}) = E[I(u_j; s_k)|s_t(u_j)], \quad k = 1, \ldots, K$$
Appendix 2

Local data exactitude

Consider an abbreviated notation as follows

\[ A^* = \text{the new proposed class value } s_{k\text{new}} \text{ at location } \mathbf{u} \]
\[ A = \text{the current class value } s_{k\text{old}} \text{ at location } \mathbf{u} \]
\[ B = \text{the neighboring class values } s_i(\mathbf{u}_j) \text{ within a template around } \mathbf{u} \]
\[ C = \text{the set of local sample data } s(\mathbf{u}_\beta) \text{ at locations } \mathbf{u}_\beta \beta = 1, \ldots, n \]

Using Bayes’ theorem the conditional probability \( \Pr\{A|B \cap C\} \) is developed as follows

\[
\Pr\{A|B \cap C\} = \frac{\Pr\{B \cap C|A\} \Pr\{A\}}{\Pr\{B \cap C\}}
\]

\[
= \frac{\Pr\{B|A\} \Pr\{C|A\} \Pr\{A\}}{\Pr\{B \cap C\}} \text{ if } B \text{ and } C \text{ are conditionally independent on } A.
\]

Then

\[
\Pr\{A|B \cap C\} = \frac{\Pr\{A^*B\} \Pr\{A\} \Pr\{A|C\} \Pr\{C\}}{\Pr\{B \cap C\} \Pr\{A\}}
\]

and the ratio of \( \Pr\{A^*|B \cap C\} \) with \( \Pr\{A|B \cap C\} \) becomes

\[
\frac{\Pr\{A^*|B \cap C\}}{\Pr\{A|B \cap C\}} = \frac{\Pr\{A^*|B\} \Pr\{A^*|C\} \Pr\{A\}}{\Pr\{A|B\} \Pr\{A|C\} \Pr\{A^*\}}
\]

The Metropolis-Hastings acceptance criterion

\[
\alpha(A, A^*) = \min \left\{ 1, \frac{\Pr\{A^*|B \cap C\} \Pr\{A\}}{\Pr\{A|B \cap C\} \Pr\{A^*\}} \right\}
\]

can now be written as follows

\[
\alpha(A, A^*) = \min \left\{ 1, \frac{\Pr\{A^*|B\} \Pr\{A|A^*\} \Pr\{A^*|C\} \Pr\{A\}}{\Pr\{A|B\} \Pr\{A^*|A^*\} \Pr\{A|C\} \Pr\{A^*\}} \right\}
\]

which is equivalent to case (14) in the text

The conditional independence assumption need not be valid. If \( B \) and \( C \) are conditionally dependent events on \( A \) then we rely on the following result

\[
\text{Cov}[I(B|A) I(C|A)] = \Pr\{B \cap C|A\} - \Pr\{B|A\} \Pr\{C|A\}
\]
where $I(B|A)$ is the indicator that the event $B|A$ occurs, i.e. $I(B|A) = 1$ if $B|A$ occurs, and zero if not. $I(B|A)$ and $I(C|A)$ are positively correlated because they relate to the same spatial variable $S(u)$, in that case

$$\Pr\{B \cap C|A\} \geq \Pr\{B|A\} \Pr\{C|A\}$$

In this paper, a power $0 < \omega(u) < 1$ is introduced such that

$$\Pr\{B|A\} (\Pr\{C|A\})^{\omega(u)} \geq \Pr\{B|A\} \Pr\{C|A\}$$

and the joint distribution $\Pr\{B \cap C|A\}$ is then modelled as

$$\Pr\{B \cap C|A\} = \Pr\{B|A\}(\Pr\{C|A\})^{\omega(u)} \quad (26)$$

This will account partially for the dependency between $B$ and $C$ and for the ”loss” in probability due to the independence assumption. If the equality (26) is included in the Metropolis-Hastings acceptance criterion

$$\alpha(A,A^*) = \min \left\{ 1, \frac{\Pr\{A^*|B \cap C\} \Pr\{A\}}{\Pr\{A|B \cap C\} \Pr\{A^*\}} \right\}$$

then one obtains

$$\alpha(A,A^*) = \min \left\{ 1, \frac{\Pr\{A|B\} \Pr\{A\}}{\Pr\{A^*|B\} \Pr\{A^*\}} \left( \frac{\Pr\{A^*|C\} \Pr\{A\}}{\Pr\{A|C\} \Pr\{A^*\}} \right)^{\omega(u)} \right\}$$

which is the equivalent to (16) in the text.
Appendix 3

Soft data integration

\[
\Pr\{A|B \cap C\} = \frac{\Pr\{A \cap B \cap C\}}{\Pr\{B \cap C\}} = \frac{\Pr\{C|A \cap B\} \Pr\{A \cap B\}}{\Pr\{B \cap C\}} = \frac{\Pr\{C|A \cap B\} \Pr\{A\|B\} \Pr\{B\}}{\Pr\{B \cap C\}}
\]

hence

\[
\frac{\Pr\{A^*|B \cap C\}}{\Pr\{A|B \cap C\}} = \frac{\Pr\{A^*|B\} \Pr\{C|A^* \cap B\}}{\Pr\{A|B\} \Pr\{C|A \cap B\}}
\]
Figure 5: True reservoir, seismic data and well data
Figure 6: Training reservoir, training seismic is obtained through a forward simulation (in this case a simple averaging process) on the training reservoir.
Figure 8: Top curve: validation error reaching minimum after 18 iterations, bottom curve: training error.
Figure 9: Four unconditional simulations. (bottom) Histogram of sand proportions of 25 realizations.
Figure 10: Four conditional simulations. (bottom) Histogram of sand proportions of 25 realizations.
Figure 11: E-type estimate and conditional variance of 25 realizations using only well data as conditioning data.
Figure 13: Four conditional simulations with angle information. (bottom) Histogram of sand proportions of 25 realizations.
Figure 14: E-type estimate and conditional variance of 25 realizations using well data and angle data as conditioning data.
Figure 15: Top curve: validation error reaching minimum after 33 iterations, bottom curve: training error.
Figure 16: (a) (top) The target seismic training data obtained from the training image versus the predicted seismic using the trained neural network (bottom) similar for the seismic validation data.
Figure 17: Four conditional simulations with soft data. (bottom) Histogram of sand proportions of 25 realizations.
Figure 18: Forward simulation of seismic on the four conditional simulation shown in Figure 7.
Figure 19: (top) average difference between the forwardly simulated seismic on 25 realizations and the true seismic data (middle) histogram of that average difference, (bottom) histogram of 25 correlation coefficients between the true seismic data and each realization.
Figure 20: E-type estimate and conditional variance of 25 realizations using well data and seismic data as conditional information.
Figure 21: Target proportion curve.
Figure 22: Four conditional simulations with the target proportion curve. (bottom) Histogram of sand proportions of 25 realizations.
Figure 23: Target proportion curve versus the proportion curve of 25 realizations.
Figure 24: E-type estimate and conditional variance of 25 realizations using well data and proportion curve.