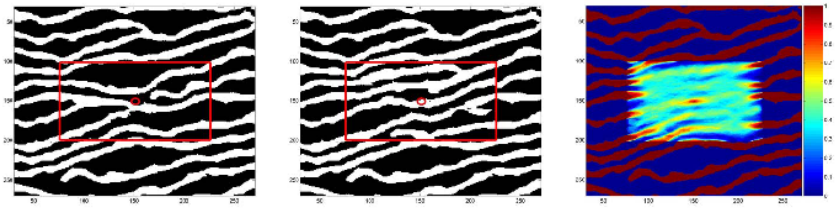
Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Note

An implementation of conditional Markov mesh simulation with parameter estimation

Overview and user's manual for version 0.1



Note no SAND/08/04

Author Marita Stien, Heidi Kjønsberg, Odd Kolbjørnsen, Petter Abrahamsen

Date 19th June 2008

© Copyright:
Norsk Regnesentral

Norsk Regnesentral
NORWEGIAN COMPUTING CENTER

Norwegian Computing Center

Norsk Regnesentral (Norwegian Computing Center, NR) is a private, independent, non-profit foundation established in 1952. NR carries out contract research and development projects in the areas of information and communication technology and applied statistical modeling. The clients are a broad range of industrial, commercial and public service organizations in the national as well as the international market. Our scientific and technical capabilities are further developed in co-operation with The Research Council of Norway and key customers. The results of our projects may take the form of reports, software, prototypes, and short courses. A proof of the confidence and appreciation our clients have for us is given by the fact that most of our new contracts are signed with previous customers.

Norsk Regnesentral
Norwegian Computing Center
Postboks 114, Blindern
NO-0314 Oslo, Norway

Besøksadresse
Office address
Gaustadalléen 23
NO-0373 Oslo, Norway

Telefon · telephone
(+47) 22 85 25 00
Telefaks · telefax
(+47) 22 69 76 60

Internett · internet
www.nr.no
E-post · e-mail
nr@nr.no

Title	**An implementation of conditional Markov mesh simulation with parameter estimation**
Author	**Marita Stien, Heidi Kjønsberg, Odd Kolbjørnsen, Petter Abrahamsen**
Date	19th June 2008
Publication number	SAND/08/04

Abstract

The main purpose of these notes is to provide documentation of version 0.1 of the Markov mesh implementation done for the Multipoint project. The report is mainly meant for internal use in the project, serving as a user's manual for running the program and as a guide for further improvements and modifications of the code.

Keywords	Markov Mesh models, sequential simulation, parameter estimation, data conditioning
Target group	All employees, MPS project partners
Availability	Open
Project	Multipoint
Project number	808002
Research field	
Number of pages	37
© Copyright Norwegian Computing Center

Contents

1 Introduction

These notes contain documentation of a prototype implementation of a Markov mesh model. The model is developed as part of a larger project for development of models for multipoint statistics, and is attractable due to its fast sequential simulation scheme.

The program implementation can perform the following tasks:

1. estimate a statistical model from a training image
2. simulate realizations from a given model
    - unconditional
    - conditioned to well(s) and/or seismic data
3. update a given realization locally
4. down scale a realization

Facies grids can be in two or three dimensions and contain multiple facies. In principle there are no limitations to the methodology, but in practice it is limited by the memory on the computer. Local updating is a post process that involves resimulation in local areas of the grid by using an McMC algorithm, and is typically used to modify existing realizations when new well data or new seismic potentials are available. Down scaling is performed to create a model on a finer level than the present realization.

In order to run the program the user needs to specify a model file containing all necessary model parameters. The parameters are explained in Section 4, with the basic theory of the applied methods being presented in Section 2. It is not crucial to read and understand the theory of Section 2 in order to properly set the input parameters, as Section 4 is attempted being self-explanatory.

The organization of the documentation is as follows. Section 2, Main methods, presents the theory related to the model construction, estimation, conditioning and the local updating. This establishes the notation. Section 3, Program structure, illustrates the main tasks enumerated above, showing how they are connected. Section 4, Input parameters, explains the input parameters needed to run the program. Section 5, Program output, describes the data, in terms of output files, that are returned by the program. A few results are for illustrative purposes included in Section 6, and a short summary and closing remarks are given i Section 7. Appendix A provides examples of typical model files.

The user can on request to the authors be provided with examples of input files such that some simple models can be run right away.

2 Main methods

This section gives a description of the main methods used in the program. It will be helpful for those interested to know the details of the various algorithms. The main parts of the section involve model specification, estimation, data conditioning and local updating.

2.1 Markov mesh models

Consider a finite, regular grid in two or three dimensions, and let the one-dimensional index i label the cells of the grid. The set of all cells is $\mathcal{G} = \{1, 2, ..., N\}$, where cell value x_i can take K different facies values, i.e. $x_i = \{0, 1, .., K-1\}$.

Markov mesh models follow a sequential path. Probabilities are constructed such that they depend on a subset of cell values from earlier in the path, and this subset is called the sequential neighborhood. Figure 1 gives an illustration of a sequential neighborhood on a two dimensional grid.

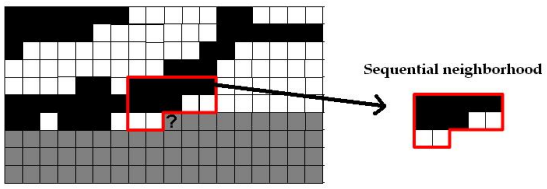


Figure 1. Illustration of sequential neighborhood. A snapshot of a simulation is displayed, and the grey cells have not yet been sampled.

We write the conditional probability for the facies at cell i as

$$\pi(x_i|x_{j<i}) = \pi(x_i|x_{\delta_i}), \tag{1}$$

where x_{δ_i} is the set of facies values for the cells in the sequential neighborhood. Markov mesh models are fully specified through the conditional probabilities in (1), i.e. the joint probability is

$$\pi(x_1, x_2, ..., x_N) = \prod_{i=1}^{N} \pi(x_i|x_{\delta_i}), \tag{2}$$

and in order to define the conditional probabilities we turn to the class of generalized linear models[1].

The concept in generalized linear models is that a linear combination of independent variables, $\boldsymbol{z}$, is linked to the expected value, μ, of the dependent variable

1. P. McCullagh and J.A. Nelder, Generalized Linear Models, Chapman & Hall, 1989.

y, through a non-linear function, $\boldsymbol{z}^T\boldsymbol{\theta} = \eta(\mu)$. We use the link function

$$\eta(\mu) = \log\left(\frac{\mu}{1-\mu}\right). \tag{3}$$

We introduce the dependent variable y_i^k,

$$y_i^k = \begin{cases} 1 & \text{when } x_i = k, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

and let $\boldsymbol{z}_i$ be a vector consisting of functions of the sequential neighbourhood of node i, and a constant term, $\boldsymbol{z}_i = [f_1(x_{\delta_i}), ..., f_P(x_{\delta_i}), 1]^T$. The functions are discussed in more detail in Section 4.7. The expected value μ can be written

$$\mu_i^k = 1 \cdot \pi(y_i^k = 1 | \boldsymbol{z}_i, \boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1}),$$

for each facies $k = 0, 1, ..., K-1$. The vectors $\boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1}$ are of size $(P+1) \times 1$ and represent coefficients for each facies corresponding to the functions in $\boldsymbol{z}_i$. Combining the above result with (3) yields the following expression for the conditional probabilities

$$\pi(y_i^k | \boldsymbol{z}_i, \boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1}) = \frac{\exp\{\boldsymbol{z}_i^T\boldsymbol{\theta}^k\}}{\sum_{k=0}^{K-1}\exp\{\boldsymbol{z}_i^T\boldsymbol{\theta}^k\}}.$$

2.2 Estimation

The model coefficients are estimated by the maximum likelihood estimator. A training image contains the data basis for the estimation. The likelihood function is given by

$$L(\boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1}; \boldsymbol{Z}, \boldsymbol{Y}) = \prod_i \frac{\prod_k \exp\{\boldsymbol{z}_i^T\boldsymbol{\theta}^k y_i^k\}}{\sum_{k=0}^{K-1}\exp\{\boldsymbol{z}_i^T\boldsymbol{\theta}^k\}},$$

where $\boldsymbol{Z}$ is an $N \times (P+1)$ matrix of the full set of observations $\boldsymbol{z}_i$, and $\boldsymbol{Y}$ is an $N \times K$ matrix of the full set of observations y_i^k for each facies. Simple derivations yield the following system of equations to be solved

$$\boldsymbol{Z}^T\boldsymbol{y}^k = \boldsymbol{Z}^T\boldsymbol{\mu}^k(\boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1}),$$

for $k = 0, 1, ..., K-1$. The vector $\boldsymbol{\mu}^k(\boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1})$ is the $N \times 1$ vector of $\mu_i^k(\boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1})$, while $\boldsymbol{y}^k = [y_1^k, y_2^k, ..., y_N^k]^T$. The above expressions are solved using the standard approach of iterated weighted least squares. For each iteration the following expressions are computed

$$\boldsymbol{\theta}_{m+1}^k = \boldsymbol{\theta}_m^k + (\boldsymbol{Z}^T\boldsymbol{W}_m^k\boldsymbol{Z})^{-1}\boldsymbol{Z}^T(\boldsymbol{y}^k - \boldsymbol{\mu}^k(\boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1}))$$

for $k = 0, 1, ..., K-1$. Here $\boldsymbol{W}_m^k = \text{diag}\{(1-\mu_1^k(\boldsymbol{\theta}_m))\mu_1^k(\boldsymbol{\theta}_m), ..., (1-\mu_N^k(\boldsymbol{\theta}_m))\mu_N^k(\boldsymbol{\theta}_m)\}$ is an $N \times N$ matrix of weights.

2.2.1 Parameter reduction

The number of coefficients to estimate is P for each facies. It is often favorable to reduce the number of parameters. We do this by removing the parameters that have least influence on the model. More specifically, the number of parameters is reduced by performing a matrix reduction of $\boldsymbol{Z}$. A matrix $\boldsymbol{Z}^*$ is the reduced matrix of size $N \times P'$ where $P' < P$ and is the product

$$\boldsymbol{Z}^* = \boldsymbol{Z}\boldsymbol{V},$$

where $\boldsymbol{V}$ is a $(P+1) \times P'$ matrix of singular vectors, which correspond to the P' largest singular values of the matrix $\boldsymbol{Z}$. The columns of $\boldsymbol{Z}^*$ are then linear combinations of the columns in $\boldsymbol{Z}$.

The set $\{\boldsymbol{\theta}^{*0}, ..., \boldsymbol{\theta}^{*K-1}\}$ is the corresponding reduced set of coefficients. We write

$$\boldsymbol{Z}^*\boldsymbol{\theta}^{*k} = \boldsymbol{Z}\boldsymbol{V}\boldsymbol{\theta}^{*k} = \boldsymbol{Z}\boldsymbol{\theta}^k.$$

Thus, the coefficients $\{\boldsymbol{\theta}^{*0}, ..., \boldsymbol{\theta}^{*K-1}\}$ are estimated using $\boldsymbol{Z}^*$, and then the true coefficients $\{\boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{K-1}\}$ are computed from

$$\boldsymbol{\theta}^k = \boldsymbol{V}\boldsymbol{\theta}^{*k}.$$

2.3 Simulation algorithm

Simulation from the Markov mesh model is performed by following the path $i = 1, 2, ..., N$ throughout the grid. For each cell the facies value is drawn according to the conditional probability $\pi(x_i|x_{\delta_i})$. It is crucial to the algorithm that when updating cell i only information about previously visited cells is taken into account, no attention is payed to any cell along the future path. The grid is scanned once, and the resulting grid configuration follows the joint probability distribution

$$\pi(x_1, x_2, ..., x_N) = \prod_{i=1}^{N} \pi(x_i|x_{\delta_i}). \tag{4}$$

2.4 Well conditioning

In the case of unconditional simulation the algorithm of Section 2.3 produces grid configurations that are consistent with the Markov mesh probability distribution (4). This is not necessarily true if the simulation is to be conditioned on well data. The reason is that update of any cell i does not condition on data located along the future path from cell i. Hence it may happen that when the simulation hits a well data point w the fixed value x_w may be inconsistent with the probability $\pi(x_w|x_{\delta_w})$.

The probability that should be used for update of x_i is $p(x_i|x_{j<i}, x_{\mathcal{W}_i})$. Here $\mathcal{W}_i$ is the set of well data along the future path from cell i; $\bigcup_i \mathcal{W}_i$ being the set of all

well data cells. We rewrite this as

$$p(x_i|x_{j<i}, x_{\mathcal{W}_i}) = \frac{p(x_i|x_{j<i}, x_{\mathcal{W}_i})}{p(x_i|x_{j<i})} p(x_i|x_{j<i}). \tag{5}$$

Equation (5) can be considered a factorization of the posterior probability into well likelihood (leftmost factor) and prior probability (rightmost factor). The unconditional Markov mesh probability is the prior information, whereas we have chosen to approximate the likelihood via indicator kriging. That is, for $p(x_i|x_{j<i}, x_{\mathcal{W}_i})$ we use the approximation

$$P(x_i|x_{j<i}, x_{\mathcal{W}_i}) = \frac{Z(x_i|x_{j<i}, x_{\mathcal{W}_i})}{Z(x_i|x_{j<i})} \pi(x_i|x_{\delta_i}) \equiv \Psi(x_i|x_{j<i}, x_{\mathcal{W}_i}) \pi(x_i|x_{\delta_i}), \tag{6}$$

where $Z(x_i|x_{j<i})$ is the predictor for x_i found by indicator kriging[2] conditioned on cells in the past of i, $Z(x_i|x_{j<i}, x_{\mathcal{W}_i})$ is the predictor conditioned also on future data points, and $\pi(\cdot)$ is the unconditional Markov mesh probability. The joint probability

$$\prod_i P(x_i|x_{j<i}, x_{\mathcal{W}_i}) \tag{7}$$

is then an approximation to the true Markov mesh model, and defines what we will refer to as the modified Markov mesh model.

Far away from any wells we expect $\Psi \approx 1$, and hence the new model based on equations (6) and (7) gives statistics similar to the prior Markov mesh model. If cell i has a non-zero correlation with a future well, the kriging function Ψ affects the overall probability P. A positive correlation implies that $Z(x_i|x_{j<i}, x_{\mathcal{W}_i})$ increases the probability for x_i to be updated to the same facies as the well, a negative correlation decreases the well's contribution to this probability. The overall effect of the well is modified by the past cells' influence on the predictors.

Simple kriging is suitable when the mean volume fraction of each facies is given. Let μ^k be the mean volume fraction of facies k, and let $\mathcal{D}$ be the set of cells on which to condition in the kriging algorithm. Our choice for how to identify $\mathcal{D}$ is explained in Section 4.8.1. Let $\boldsymbol{\Sigma}$ be a matrix whose elements are the covariances between the different data points $j_1, j_2 \in \mathcal{D}$, $\boldsymbol{\Sigma}_{j_1,j_2} = \text{Cov}(j_1, j_2)$; and $\boldsymbol{c}$ is a vector of the covariance functions between the fixed update location i and the data points, $\boldsymbol{c}_j = \text{Cov}(i, j), j \in \mathcal{D}$. Define in addition the $n \times 1$ vectors $\hat{\boldsymbol{y}}^k = [y^k_{j_1}, y^k_{j_2}, ..., y^k_{j_n}]^T$ and $\boldsymbol{e} = [1, 1, ..., 1]^T$, where n is the number of cells in $\mathcal{D}$, and

$$y^k_j = \begin{cases} 1 & \text{when } x_j = k, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

The predictor for facies k is then found via the linear expression

$$Z(x_i = k|x_{\{j\in\mathcal{D}\}}) = \mu^k + \boldsymbol{c}^T \boldsymbol{\Sigma}^{-1} (\hat{\boldsymbol{y}}^k - \mu^k \boldsymbol{e}). \tag{8}$$

2. A.G. Journel, *Nonparametric Estimation of Spatial Distributions*, Math. Geol. 15(3), 1983.

For each cell i this expression is used to find the predictors $Z(x_i|x_{j<i}, x_{\mathcal{W}_i})$ and $Z(x_i|x_{j<i})$ of (6).

2.5 Seismic conditioning

If seismic data is present this should also contribute to Eq. (5). We assume that the seismic data is available in the form of a full facies probability cube, i.e. that the seismic probabilities $p(x_i|m)$, with m being the seismic raw data, are known for all cells i in the grid. Equation (5) asks for the likelihood, and we therefore calculate

$$l(x_i) \equiv \frac{p(x_i|m)}{p(x_i)} \propto p(m|x_i), \tag{9}$$

where $p(x_i|m)$ is taken from the seismic facies probability cube and $p(x_i)$ is the marginal facies probability found from the training image. A weight factor β for the seismic potential is manually included, which allows for some user's adjustment of the importance of the seismic. This is common procedure when taking into account seismic data. With seismic data being present in addition to well data the adjusting factor Ψ of (6) thus is replaced by

$$\Psi(x_i|x_{j<i}, x_{\mathcal{W}_i}) \to l(x_i)^\beta \Psi(x_i|x_{j<i}, x_{\mathcal{W}_i}), \tag{10}$$

with $\beta \in [0, 1]$. If no well data is present $\Psi(x_i|x_{j<i}, x_{\mathcal{W}_i}) = l(x_i)^\beta$.

2.6 Local update

The modified Markov mesh model (7) is well suited for iterations. A main motivation for establishing an iterative method is to render possible local update in an already existing grid configuration, typically as a result of new well data or modified seismic potentials. Iterations could also be used to clear unwanted kriging effects during the initial establishment of a reservoir configuration, to ensure that it is consistent with the statistics of the prior model. We propose to use a Metropolis Hastings algorithm, where proposal configurations are established via block update based on the modified Markov model (7), and the accept probability ensures that sampling is done from the prior model conditioned on data. In the following we describe one step in the Markov chain of the Monte Carlo simulation.

Let ν be a label for the existing configuration of the grid, i.e. the grid configuration is x_ν. The proposal configuration will be denoted μ. Pick according to some rule a connected set of cells $\mathcal{B} \subseteq \mathcal{G}$, where $\mathcal{G}$ is the full grid. If the set $\mathcal{B}$ includes data cells or cells that for some other reason are supposed to have fixed facies throughout the iterations, let the set of these cells be denoted $\mathcal{B}_0$. Let $\mathcal{B}_1 = \mathcal{B} \setminus \mathcal{B}_0$, and define $\partial\mathcal{B}_1$ as the set of cells in $\mathcal{G} \setminus \mathcal{B}_1$ that according to the prior model may be affected by a change in the set $\mathcal{B}_1$. That is, $j \in \partial\mathcal{B}_1$ iff $\exists k \in \delta_j : k \in \mathcal{B}_1$. Then

$(\mathcal{B}_0 \cup \partial\mathcal{B}_1) \cap \{j : j > i\}$ is the union of the set of future cells from a fixed location i that are within $\mathcal{B}$ and that we want to condition on, and the set of future cells from position i that according to the prior model are affected by a change in the set $\mathcal{B}_1$. The new configuration μ is established as follows: scan through the part of the simulation path that is within $\mathcal{B}_1$, and for each cell $i \in \mathcal{B}_1$ draw its new facies value x_i for configuration μ from the probability

$$\Psi(x_i|x_{j<i}, x_{\mathcal{W}_i \cup \{(\mathcal{B}_0 \cup \partial\mathcal{B}_1) \cap \{j:j>i\}\}})\pi(x_i|x_{\delta_i}); \tag{11}$$

for all cells in $\mathcal{G} \setminus \mathcal{B}_1$ let the facies be as in the state ν. This defines the proposal configuration μ.

Acceptance or rejection of the configuration ν is done with probability

$$\alpha = \min\left(\frac{q_\nu \pi(x_\mu)}{q_\mu \pi(x_\nu)}, 1\right), \tag{12}$$

where q_μ is the probability for suggesting the new configuration μ, starting from ν, q_ν is the probability for suggesting the old configuration ν, starting from μ, and $\pi(x_\mu)$ and $\pi(x_\nu)$ are the prior Markov mesh probabilities. Since all suggested states conform with data, this accept probability ensures that sampling is done from the prior model conditioned on data. The accept probability can be rewritten to

$$\alpha = \min\left(\left(\prod_{i \in \mathcal{B}_1} \frac{\Psi(x_i)|_\nu}{\Psi(x_i)|_\mu}\right)\left(\prod_{i \in \mathcal{B}_0 \bigcup \partial\mathcal{B}_1} \frac{\pi(x_i)|_\mu}{\pi(x_i)|_\nu}\right), 1\right). \tag{13}$$

In the notation of equation (13) the conditional dependencies in the function arguments are suppressed for readability.

3 Program structure

The main task supported by the present implementation are:

- Based on an input training image, estimate parameters for a Markov mesh model.
- Create a number of independent realizations from the Markov mesh model by using sequential simulation.
- Each realization can be conditioned on wells and/or seismic data.

In addition, the program can

- Do local update on an existing realization.
- Instead of estimation and subsequent sequential simulation the user can provide a grid realization on a coarse scale, then ask the program to fill in the grid on a finer scale.

Figure 2 illustrates the logical structure of the program from a user's perspective. Parameters that control the behavior of the program are described in the next section.

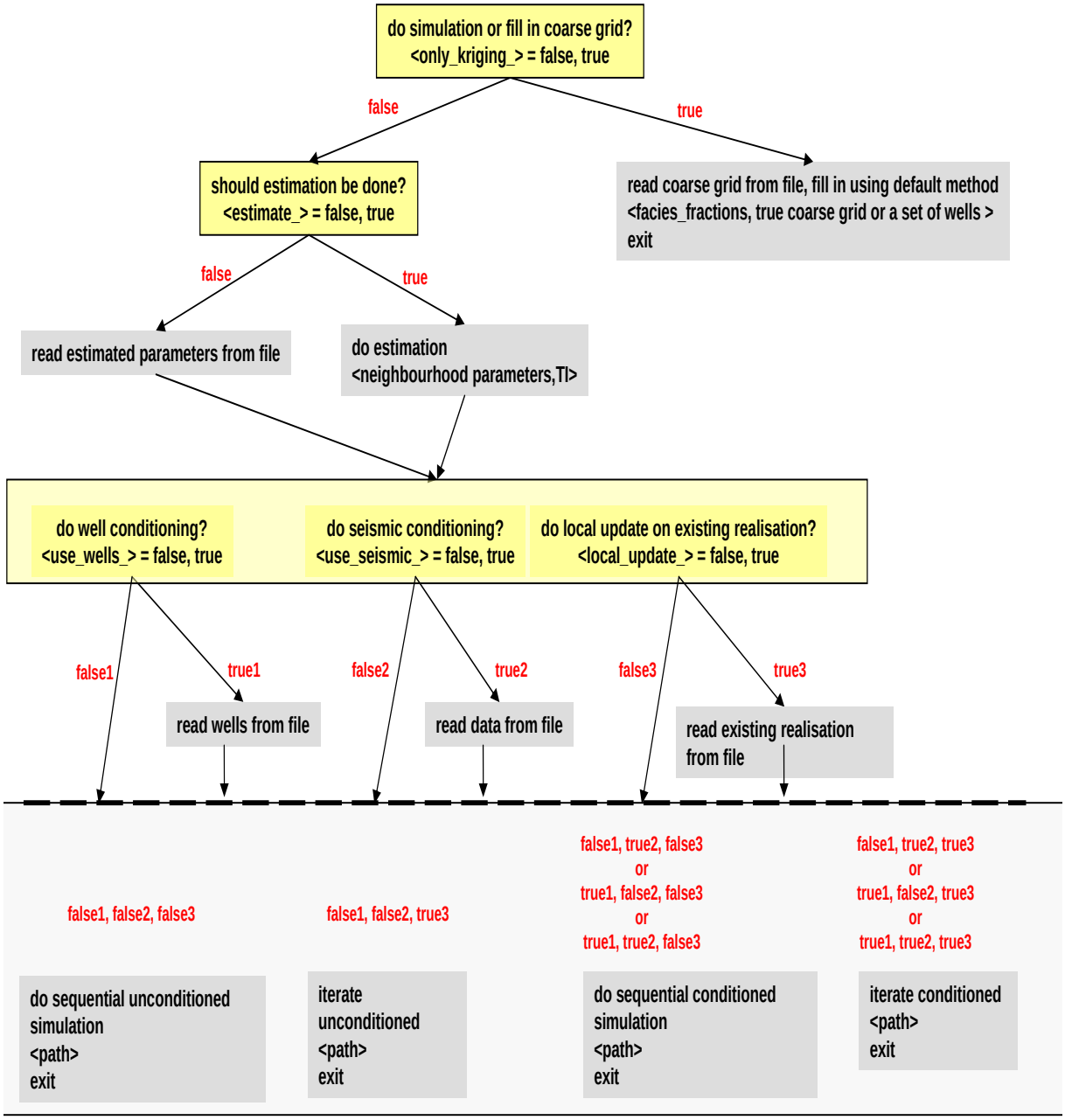


Figure 2. Graphical illustration of the program structure.

4 Input parameters

In this section we explain how to set the input parameters in the model file. We use a model file structure which is divided into commands. Each command starts with the command name written in capital letters, and ends with a semicolon (;). The input parameters to be specified by the user are written between the command name and the semicolon. The exclamation mark (!) is used in front of comments, i.e. everything written on the line behind an exclamation mark is not read by the program.

In some cases input files in addition to the model file are needed by the program. These must all be located on the root directory of the program.

4.1 Main flow parameters

The first parameters to be set determine which workflow to run, using command `WORKFLOW`. We refer to Figure 2, which shows various workflow actions and how one follows the other. There are five boolean parameters to be set, where 1 means true and 0 false.

1. `estimate`: Set to *true* if the coefficients $\boldsymbol{\theta}$ should be estimated or *false* if the coefficients should be read from an existing file. Further parameter input is described in sections 4.4, 4.5, 4.6, and 4.7.

2. `only_kriging`: Set to *true* if coarse grid values from file should be filled into a finer grid. The coarse grid values must be available via file. See Section 4.11 for further details. If the parameter is set to *false* sequential simulation is run, either with or without conditioning. See sections 4.2, 4.3, 4.4, and 4.5 for further parameter settings.

3. `use_wells`: Set to *true* if wells should be taken into account, and *false* if there is no well conditioning. Further parameter input is described in Section 4.8.

4. `use_seismic`: Set to *true* if seismic should be taken into account, and *false* if there is no seismic conditioning. Seismic input information is described in Section 4.9.

5. `do_iterate`: Set to *true* if local update is to be performed. If so, the initial grid must be provided in a file named `initialgrid.txt`. Further parameter input is described in Section 4.10. If *false* then no local update is to be done.

Example of workflow parameters:

```
WORKFLOW
0  !estimate_       1 = true  0 = false
0  !only_kriging_   1 = true  0 = false
1  !use_wells_      1 = true  0 = false
1  !use_seismic_    1 = true  0 = false
0  !do_iterate_     1 = true  0 = false
;
```

4.2 Realizations

The number of realizations, the output file format for writing the grid configurations to file, and the location and filename prefix for the realizations must be given under the `REALIZATIONS` command in the model file.

There are two options for the file format, either a header is included or not. A header must be included if the realizations are to be run through the analysis program *FaProp*[3], otherwise only facies values from 0 to $K-1$ are printed. The program adds realization number and a *.txt* ending to the output prefix.

The prefix is also used for all other output files from the program. If `Output prefix` ends with backslash (\) it refers only to the output folder, and all output files will be given their default names (see Section 5). The file folder must be manually created before simulation.

```
REALIZATIONS
10                          !Number of realizations
1                           !Header 1 = true  0 = false
C:\MarkovMeshSimulations\realization  !Output prefix
;
```

4.3 Simbox definition

The simbox definitions, command `SIMBOXDEFINITION`, are for defining the area for simulation. Only two-dimensional rectangles or three-dimensional boxes are supported. The user must set the corner point `x0` , `y0` and `z0` which is the upper left corner at the top of the cube. Cell sizes are given by the parameters `dx`, `dy` and `dz`, and the number of cells by the parameters `nx`, `ny` and `nz`. Padding to avoid the edge effects is included in the program with default parameters. Note that for two-dimensional grids `nz` is set to 1, see example:

3. H. Soleng, A.R. Syversveen and O. Kolbjørnsen, *Comparing Facies Realizations - Defining Metrices on Realization Space*, Proceedings of the 10th European Conference in the Mathematics of Oil Recovery, 2006

```
SIMBOXDEFINITION
0 0 0     !x0, y0, z0
1 1 1     !dx, dy, dz
100 100 1 !nx, ny, nz
;
```

The example above describes a simulation rectangle in two dimensions, where the upper left corner is located at the origin. There are 100 cells in each direction and they are all of size 1^2.

4.4 Training image

The training image must be either a 2D rectangle or a 3D box. Under the command `TI`, the first input to be specified is the name and location of the training image file. The file format is a text file with no header, containing only the facies values. The values are separated either by space or line break, and the order of cell listings are fastest in x direction and slowest in the z direction. Next, the size of the training image must be given, see example:

```
TI
C:\TrainingImages\channelTI.txt       !Directory and filename of TI
40 40 50                             !X, Y and Z dimension
;
```

4.5 Direction - Path

The program supports three main simulation paths, determined by the command `DIRECTION`. The first main path is the usual left to right - top to bottom simulation path. The second path is from left to right and right to left for every other row for each vertical layer. Each of these two paths is suitable for sequential simulation. The third choice of path is a random path, which is useful for the `kriging_only` version of the program (see Section 4.1).

In addition, there is a fourth path choice. This choice is useful for local update. The subpath traversing the cells to be updated then varies randomly between a left to right - top to bottom simulation path (NorthWest), and a right to left - bottom to top (SouthEast) simulation path in each vertical layer. The vertical layers are traversed from top to bottom (TOP).

```
DIRECTION
!0    !One way simulation
!1    !Back and forth simulation, top to bottom
!2    !Random
3     !Subpaths vary between NW_TOP and SE_TOP
;
```

Be aware that regardless of simulation path, only one set of parameters is estimated. This parameter set refers to the one way simulation path. The use of any other path specification under `DIRECTION` implicitly assumes a symmetric model.

4.6 Facies

The number of facies in the training image and realizations must be given as an input. If the `only_kriging` version of the program is being used, also the volume fractions of the facies must be provided. Volume fractions are in all other cases automatically calculated from the training image.

```
FACIES
2       !number of facies
0.723   !volume fraction facies 0, used iff only_kriging == 1
0.277   !volume fraction facies 1, used iff only_kriging == 1
;
```

4.7 Neighborhood

If the `estimate` parameter is set to *true*, neighborhood parameters must be provided. If `estimate` is set to *false* and the parameters are needed for simulation (`only_kriging` is set to *false*), they are automatically read from the coefficient file, see Section 5.

We have divided the neighborhood into two categories; two-point interactions and higher-point interactions under the command names `TWOPOINT` and `NEIGHBORHOOD`, respectively. The explanatory variables in the conditional probabilites are functions of neighborhood variables, i.e.

$$\boldsymbol{z}_i = \{f_1(x_{\delta_i}), f_2(x_{\delta_i}), ..., f_P(x_{\delta_i}), 1\}.$$

In the following two sections we explain how the $f(\cdot)$ functions have been chosen and which input ranges that are to be determined by the user.

4.7.1 Two-point interactions

For two-point interactions we consider the strength between the variables x_i and all the variables in the two-point interaction neighborhood. For each facies k we assign an indicator function $f^k(x_j^{2point})$ which yields

$$f^k(x_j^{2point}) = \begin{cases} 1 & \text{if } x_j^{2point} = k \\ 0 & \text{otherwise,} \end{cases}$$

where x_j^{2point} is a variable in the two-point interaction neighborhood, j labeling the cell.

Figures 3(a)-(c) display the variables included in the two-point interaction neighborhood. The figures show one plane at the time, where l_x, l_y and l_z is the

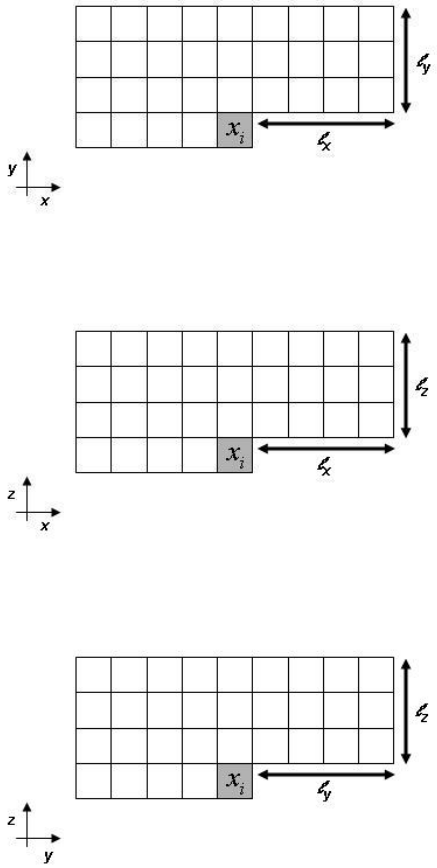


Figure 3. Illustration of the two-point interaction neighborhoods for all three planes (a) x-y plane, (b) x-z plane and (c) y-z plane.

extension of the neighborhood in the corresponding direction. Note that we only consider variables from the planes orthogonal to eachother. For the total number of N^{2point} cells in the two-point interaction neighborhood of cell i, we get the following set of explanatory variables

$$z_i^{2point} = \{f^1(x_1^{2point}), ..., f^K(x_1^{2point}), ..., f^1(x_{N^{2point}}^{2point}), ..., f^K(x_{N^{2point}}^{2point})\}.$$

In the model file the user must give the parameters l_x, l_y and l_z, for instance:

```
TWOPOINT
2 2 ! XY plane lx ly
1 1 ! XZ plane lx lz
1 1 ! YZ plane ly lz
;
```

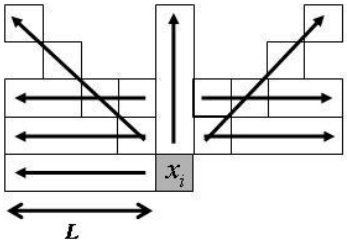


Figure 4. Illustration of the strips of cells where higher-point interactions are considered. Arrows indicates the directions and in which order the number of interaction terms increases.

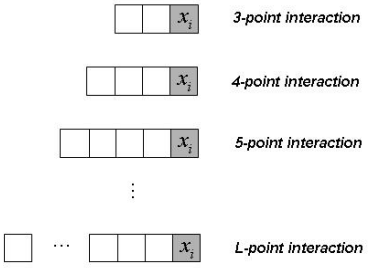


Figure 5. Example of a strip of cells and the interactions that are included in the explanatory variables.

4.7.2 Higher-point interactions

In addition to two-point interactions, three- to L-point interactions are also implemented. The size of L must be given by the user for each direction x, y and z. Not all possible higher-point interactions within a certain neighborhood range are considered. Certain strips of cells are selected, see Figure 4 where each strip in one plane is indicated by arrows. All three planes are similar. Four additional strips are also included, going from the center cell i diagonally out one step in each direction.

Figure 5 shows for one of the strips which interactions that are taken into account. We let x_j^{lpoint} be the set of cells included in a l-point interaction for strip j, with the indicator function

$$f^k(x_j^{lpoint}) = \begin{cases} 1 & \text{if all cells in } x_j^{lpoint} = k \\ 0 & \text{otherwise,} \end{cases}$$

For the total number of R strips rooted at cell i, we get the following sets of ex-

planatory variables

$$
\begin{aligned}
z_i^{3point} &= \{f^1(x_1^{3point}), ..., f^K(x_1^{3point}), ..., f^1(x_R^{3point}), ..., f^K(x_R^{3point})\} \\
z_i^{4point} &= \{f^1(x_1^{4point}), ..., f^K(x_1^{4point}), ..., f^1(x_R^{4point}), ..., f^K(x_R^{4point})\} \\
&\vdots \\
z_i^{Lpoint} &= \{f^1(x_1^{Lpoint}), ..., f^K(x_1^{Lpoint}), ..., f^1(x_R^{Lpoint}), ..., f^K(x_R^{Lpoint})\}
\end{aligned}
$$

In the model file the user must give the parameter L for the various directions, both along the three main axes and the diagonal lengths, for instance:

```
NEIGHBOURHOOD
2 !X direction
2 !Y direction
1 !Z direction
1 !XY 2D diagnoals
1 !XZ 2D diagonals
1 !YZ 2D diagnoals
1 !XYZ 3D diagonals
;
```

4.8 Well conditioning

There are two aspects of the kriging algorithm for hard data conditioning (wells) that are controlled by parameters: the correlation structure used to set up Σ and c, see Eq. (8), and the number of cells from which the predictor is calculated. These parameters are set under the model file command `CONDITIONING`. In addition the locations and facies of wells must be provided via an input file that is specified under the command `WELLDATA`.

4.8.1 Correlation structure and shell search

The parameters that determine the kriging details, command `CONDITIONING`, are specified in the model file as

```
CONDITIONING
1  !variogram type !0 = spherical, 1 = empiric
30 !variogram range Rx (empiric), variogram range (spheric)
30 !variogram range Ry (empiric), variogram subrange (spheric)
10 !variogram range Rz (empiric), variogram angle (spheric)
1  !N, the max number of kriging neighbours
1  !N1, the max number of kriging neighbours forward, used in iterations
1  !N2, the max number of kriging neighbours backward, used in iterations
;
```

In the following we describe the details of the algorithms controlled by the parameters, and thereby explain what each parameter means.

Correlation structure

Two different variogram types are supported by our implementation: spherical variogram and empiric variogram. The choice of variogram type is controlled by the first parameter listed under `CONDITIONING`. If empiric variogram is chosen, the program automatically calculates the variogram in the x, y and z-direction from the input training image (see Section 4.4). The spherical variogram is all synthetic, with (2D) range, subrange and angle relative to the x-axis specified as input parameters. The variograms are subsequently used for the correlations that enter the kriging matrix and kriging vector, with elements given by $\mathbf{\Sigma}_{j_1,j_2} = \text{Cov}(j_1, j_2)$ and $\boldsymbol{c}_j = \text{Cov}(i, j)$, respectively.

The meaning of the three next parameters under `CONDITIONING` depends on the choice of variogram type. If empiric variogram is chosen, the parameters denote range R_x in x-direction, range R_y in y-direction, and range R_z in z-direction. For any fixed cell i these parameters define a box centered at cell i, with side lengths $2R_x + 1$, $2R_y + 1$, and $2R_z + 1$. This box defines the cells that are candidates for being used as conditioning cells when doing kriging for cell position i. Cells inside the box are taken into account provided they fulfil various criteria (explained below), cells outside the box are not taken into account. Hence the ranges R_x, R_y and R_z should have values that are roughly the same as the correlation lengths of the TI. The output file `correlations.txt` (see Section 5) contains the correlations of the training image, and can be used to identify appropriate values for R_x, R_y and R_z.

If spherical variogram is chosen the three parameters signify (2D) range, subrange and angle relative to the x-axis. From these parameters the program automatically calculates a box that encompasses the variogram range. This box defines cells that are considered candidates for playing the role as kriging data.

Shell search

To identify the cells $\mathcal{D}$ used to calculate the predictor $Z(x_i = k|x_{\{j\in\mathcal{D}\}})$ in Equation (8), the basic algorithm is a shell search algorithm. That is, we search outwards from cell i in cubic shells (if 2D in quadratic shells) and identify in each shell the cells that either are future wells or have already been simulated, though with some minor requirements that are listed below. We continue until the shell is entirely outside the box defined by the range parameters described above, or until the number of found cells reaches a set limit, whichever occurs first.

The details of the shell search algorithm differs slightly depending on the flow parameters of the program, see Section 4.1.

- `only_kriging` == 1 (filling in a coarse grid): In each shell of the shell search the program identifies cells that are either future wells or have been filled in before. The single parameter `N` under `CONDITIONING` defines the upper limit for how many cells to include.

- `only_kriging` == 0, `use_wells` == 1, `do_iterate_` == 0 (sequential simulation with hard data conditioning): Before shell search is started, the program accepts as kriging data future wells that are inside the box defined by the variogram range parameters. If the number of these future wells exceeds `N`, a subset of the wells is randomly chosen. If the number of wells is lower than `N` the shell search algorithm starts, and previsouly simulated cells are included until the limit `N` is reached or shell search is beyond variogram range.

- `only_kriging` == 0, `use_wells` == 1, `do_iterate` == 1 (local update of the initial configuration): Two parameters are used for the iterative phase of the simulation: `N1` and `N2` are maximal limits for the number of conditioning cells along the forward and backward path, respectively. Shell search is used for each direction independently.

4.8.2 Well input file

Well positions and well facies must be specified in an input file. The name of the file is specified as in this example:

```
WELLDATA
welldata.txt  !Name of the welldata file
;
```

The file format is, assuming n well positions with facies values $k_1, k_2, ..., k_n$:

```
x1 y1 z1 k1
x2 y2 z2 k2
x3 y3 z3 k3
...
xn yn zn fn
```

The coordinates x, y, z must refer to the same unit system as is used to define the simulation box, see Section 4.2.

4.9 Seismic conditioning

If seismic conditioning is to be done the name of a file specifying the seismic probability cube must be provided. In addition the seismic weight factor must be set, see Eq. (10). The following example illustrates how to set the seismic parameters:

```
SEISMICATTRIBUTES
0.5                    !0 <= seis_factor_ <= 1
seismicattributes.txt  !Name of the seismic attributes file
;
```

The file format is, assuming K facies values:

```
p(x1 = 0) p(x2 = 0) ... p(xN = 0)
p(x1 = 1) p(x2 = 1) ... p(xN = 1)
...
p(x1 = K-1) p(x2 = K-1) ... p(xN = K-1)
```

Here $x_1, x_2, ..., x_N$ are facies variables, one for each cell of the grid. N is the total number of grid cells listed in the file, and this number must be consistent with the simbox definitions of Section 4.3. The indices $1, 2, ..., N$ label the cells according to the ordering: first run over x, then over y, last over z.

4.10 Local update

Local update requires two commands to be specified in the model file: `LOCALUPDATE` and `ITERATIONS`. The first command simply specifies the name of the required input file listing the cells where update is allowed to happen, the second command specifies the number of iterations and the possible sizes of the update boxes used during the iterations.

The input file must be specified as in this example:

```
LOCALUPDATE
maskslocalupdate.txt    !Name of the local update mask file
;
```

The file format is, assuming n cells are allowed updated:

```
x1 y1 z1
x2 y2 z2
x3 y3 z3
...
xn yn zn
```

The coordinates x, y, z must refer to the same unit system as is used to define the simulation box, see Section 4.2.

Input parameters that control the iteration algorithm are specified as follows:

```
ITERATIONS
1000   !the number of iterations if do_iterate_ == true
25     !lxmin
25     !lymin
0      !lzmin
50     !lxmax
50     !lymax
50     !lzmax
```

The first parameter describes the number of iteration steps that are to be carried out. The remaining parameters are understood as follows: Each step of the Markov chain that defines the iterative process uses a connected set of cells $\mathcal{B} \subseteq \mathcal{G}$, where $\mathcal{G}$ is the full grid. See Section 2.6 for a detailed description of the theory behind the method. The present implementation uses a rectangular box for $\mathcal{B}$. The six parameters `lxmin`,..., `lzmax` defines this box according to the following algorithm, which is used once for each step of the Markov chain:

1. Pick randomly one of the cells that are allowed to change facies during the iterative phase, i.e. a cell from the set $\mathcal{C}$ specified by the input file's `LOCALUPDATE`, provided the cell is not a well.

2. Pick randomly the size parameters l_x, l_y and l_z, such that each parameter is in the interval $l \in [\texttt{lmin}, \texttt{lmax}]$.

3. Define a box with center in the picked cell, and sides of lengths $2l_x + 1$, $2l_y + 1$ and $2l_z + 1$. The set of cells within the box constitute $\mathcal{B}$.

4. The set of cells that are updated during this iteration step is given by $\mathcal{B}_1 \cup \mathcal{C}$.

4.11 Fill in from coarse grid

If the main flow parameter `only_kriging` is set to *true*, coarse grid values from file should be filled into a finer grid. The coarse grid values must be available in a file called `coarsegrid.txt`. The format of this file should be identical to the header-less output file from sequential simulation (see sections 4.2 and 5), and the number of values should in 3D be 1/8 of the size of the present simbox (see Section 4.2). The program will krige unsampled cells in the finer grid.

5 Program output

Depending on the main flow parameters the program produces various output information in the form of files. The names of these files all start with the prefix specified in the command `REALIZATIONS`. All relevant file directories must be set up before simulation. Apart from the prefix, the output file names are:

- If `estimate` == *true*:
  `coeff.txt`: Estimated model parameters. The file header contains the sequential neighborhood parameters.

- If `only_kriging` == *true*:
  `0.txt`: Resulting configuration of the fine grid.

- If `only_kriging` == *false* and `use_wells` == *false/true* and `use_seismic` == *false/true*:
  `<realization number>.txt`: Resulting grid configuration for the indicated realization number.

- If `use_wells` == *true*:
  `correlations.txt`: Correlations of the training image.

- If `do_iterate` == *true*:
  `<realization number>.txt`: Final grid configuration for the indicated realization number.

  `<realization number>_<iteration number>after.txt`: Intermediate grid configurations, written every 10th iteration.

  `<realization number>_faciesCount_n<iteration number>_N<total number of iterations>.txt`: Facies count for each facies for each cell. Written every 1000th iteration and at the end of the iterations. The listing is according to the rule: first run over cell index, then run over facies.

  `accept.txt`: Statistics for the accept rate. Updated each 1000th iteration and at the end of the iterations, each line being on the format `[<realization number> <number of tried iterations> <number of accepted iterations>]`.

In addition, for future reference all relevant input files are automatically copied to the chosen output directory. The possible files are: `model.txt`, `coeff.txt`, `welldata.txt`, `seismicattributes.txt`, `initialgrid.txt`, and `maskslocalupdate.txt`.

6 Results

This section shows a few results that have been obtained by using the program. The main objective is to illustrate possible usage, not to discuss in detail the validity of the results. All illustrations were created from the data of the program's output files using MatLab.

Unconditional simulation

Figure 6 displays two different 2D training images, one characterized by channels, the other by more irregular sand objects. These training images have been used to estimate model parameters as described in these notes, and hence define two different Markov mesh models. These Markov models have in turn been used for simulation, be it unilateral unconditional or conditional, or local update.

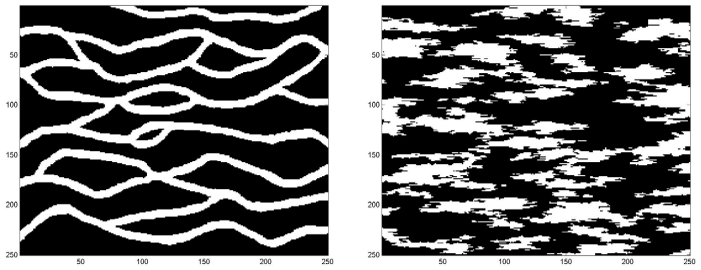


Figure 6. Two training images.

Figure 7 shows one random sample for each model. Visual comparison to the training images shows that the implemented method for parametrization and parameter estimation in these cases is able to reproduce the characteristics of the training image quite well.

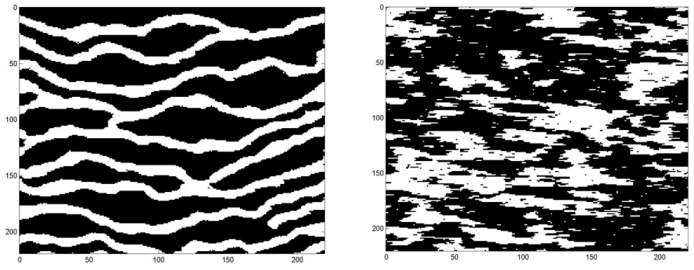


Figure 7. Results from unconditional simulation of estimated Markov mesh model.

Unilateral conditioned simulation

Figure 8 shows two random samples from conditional simulations using the modified unilateral Markov mesh model of Equation (7). Conditioning is done with respect to a well in the middle of each figure. Comparing these two samples with the samples from the unconditional Markov mesh models, see figure 7, there is no observable statistical difference between the samples of the prior models and the modified models. The only difference is that the modified Markov mesh models locate sand objects correctly around the well position.

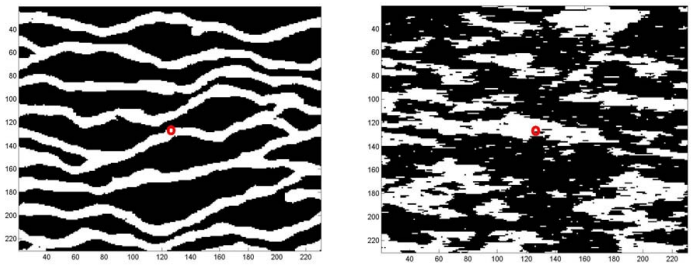


Figure 8. Results from conditional simulation using indicator kriging and estimated prior model. Conditioned on object facies well in cell the center of the figure, the well being indicated with a circle.

We can investigate the statistical properties in more detail by simulating many realizations from one model, and then summarize the results. In particular, the statistics of the modified Markov mesh model can be compared to the statistics resulting from rejection sampling of realizations created by the prior Markov mesh model. The approximation done when using indicator kriging is to be considered satisfactory if rejection sampling and modified Markov model give the same statistical results.

Figure 9 displays the conditional marginal probabilities, for the channel model, conditioned on object well (top row) and background well (bottom row). In each case the well is located in the middle of the grid. The results from rejection sampling are shown in the left column. The right column displays the results of conditioned simulation using the modified Markov mesh model with indicator kriging. Comparison to the rejection sampling gives the impression that the overall results of the modified Markov mesh model are satisfactory.

Figure 10 displays results for conditioning on a channel edge, i.e. a transition from an $x = 0$ well to an $x = 1$ well. Comparison to rejection sampling (left) shows that the main features are well reproduced by the modified model (right). In particular the results are good close to the wells.

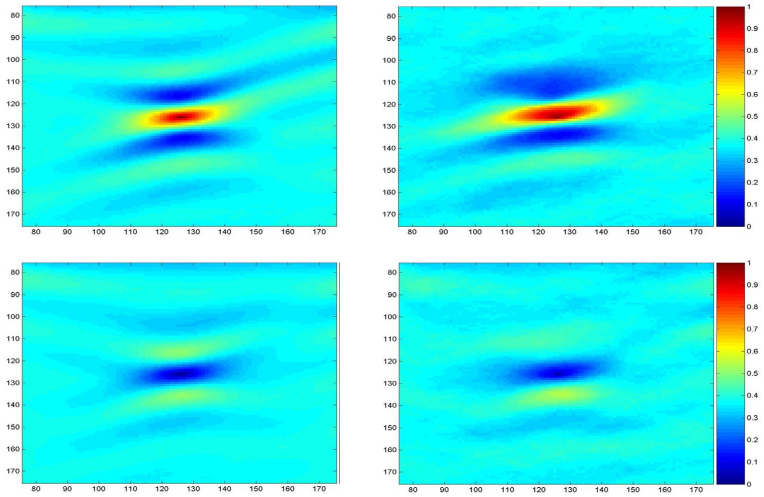


Figure 9. Conditional marginal statistics, displayed as the cell-wise probability $p(x_i = 1)$. Left column: rejection sampling from prior; right column: indicator kriging with respect to the well and two cells in the past path. Top row: well has facies $x = 1$; bottom row: well has facies $x = 0$.

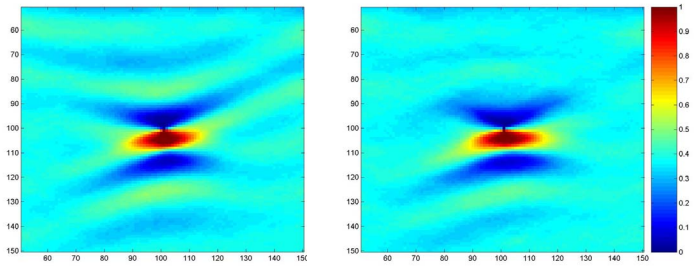


Figure 10. Conditioning on two neighboring wells, one with $x = 0$ in position (101,100), the other with $x = 1$ in position (101,101). Cell-wise probability $p(x_i = 1)$. Left: rejection sampling; Right: Modified unilateral Markov mesh model.

Local Update

The iterative model can be used to perform local update of an existing realization. We illustrate this in Figure 11. The left hand side of the figure shows an example of an initial grid configuration created by unconditional simulation. This realization is to be updated with respect to a new well in grid position (126, 126). The well position is indicated with a circle at the center of the figure. The value of the well is supposed to be $x = 1$, which is not in agreement with the initial grid configuration. Local update is to be carried out in an area $\mathcal{A}$ defined by the rectangular box shown in red in the initial grid. All cells outside $\mathcal{A}$ are fixed. The iterative method, using block update, is now applied to this problem. The region $\mathcal{B}$ used to generate a suggestion state for the Metropolis-Hastings algorithm is for each element of the Monte Carlo chain a randomly drawn rectangle, and the cells

that satisfy $i \in \mathcal{B} \cap \mathcal{A}$ are updated according to the algorithm.

The middle grid of Figure 11 displays a snapshot of the grid during the iterative process. The channels have now been adjusted such that they are consistent with the well data. In addition the channels nicely match the fixed part of the grid outside the marked rectangle.

At the right hand side of Figure 11 we display the cell-wise statistics of the grid after 3000 iterations. The figure illustrates that along the edges of the update rectangle $\mathcal{A}$ the probabilities follow the conditioning provided by the cells outside the rectangle. Also conditioning to the well is good. The other areas of the local update rectangle provide evidence that the mixing of the iterative process is satisfactory. This conclusion follows from the fact that the cell-wise statistics tend to smear out and match the marginal probability $p(x = 1)$, only broken by areas with a higher channel probability, areas that to a large extent obviously follow from the edge and well conditioning.

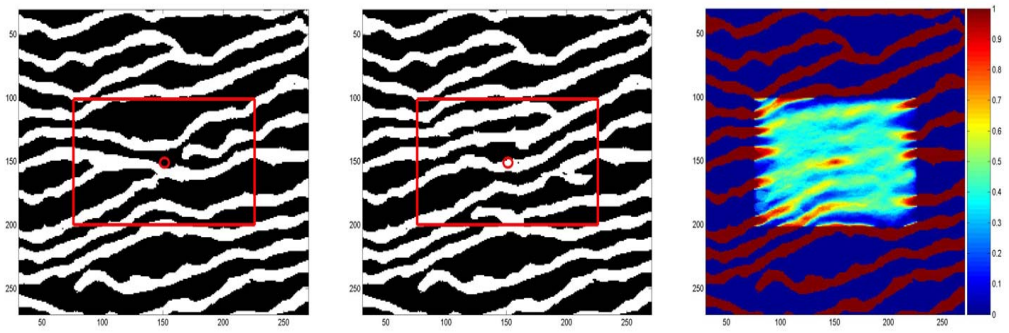


Figure 11. Left: Initial grid configuration before local update; middle: snapshot of the grid configuration during iterative process; right: cell-wise probability $p(x_i = 1)$ after 3000 iterations. The well position is indicated, but not conditioned to, in the initial grid. The iterative process conditions on the well, which has facies $x_w = 1$. The rectangle marks the area of local update.

7 Summary

These notes provide documentation for a prototype implementation of a Markov mesh model. The implementation can estimate a statistical model from a training image; simulate realizations from a given model, either unconditioned or conditioned to well(s) and/or seismic data; update a given realization locally; or down scale a realization.

The theoretical formulation of the implemented methods is provided, as well as a detailed overview of the program's input parameters and main work flows. A few results were included for illustrative purposes.

A Examples of model files

We give here two examples of model files. In the first example the program is supposed to estimate parameters based on a training image, then make 100 realizations conditioned on well data. In the second example we show the model file that will take a grid realization as input, then perform local update on this realization. In reality, a likely situation is that Example 2 is carried out some time after Example 1, and that one of the realizations created by Example 1 is desired updated due to new well data.

Example 1

```
WORKFLOW
1  !estimate_
0  !only_kriging_
1  !use_wells_
0  !use_seismic_
0  !do_iterate_
;
REALIZATIONS
100            !Number of realizations
0              !Header 1 = true 0 = false
C:\Project\    !Output directory and prefix
;
SIMBOXDEFINITION
0 0 0  !x0, y0, z0
1 1 1  !dx, dy, dz
300 300 1 !nx, ny, nz
;
TI
C:\TrainingImages\channelTI.dat
250 250 1                        !Size x y z direction
;
DIRECTION
0      !One way simulation
!1     !Back and forth simulation, top to bottom
!2     !Random
!3     !Main path is one way, subpaths vary between NWTOP and SETOP
;
FACIES
2       !number of facies
```

```
;
NEIGHBOURHOOD
8 !X
8 !Y
1 !Z
1 !XY Diagnoals
1 !XZ Diagonals
1 !YZ Diagnoals
1 !XYZ Diagonals
;
TWOPOINT
6 6 ! XY plane lx ly
1 1 ! XZ plane lx lz
1 1 ! YZ plane ly lz
;
CONDITIONING
1     !variogram type !0 = spherical (2D), 1 = empiric (2D and 3D)
30    !variogram range Rx (empiric), variogram range (spherical)
30    !variogram range Ry (empiric), variogram subrange (spherical)
10    !variogram range Rz (empiric), variogram angle (spherical)
3     !maximum number of kriging neighbours
!1    !maximum number of kriging neighbours forward, used during iterations
!1    !maximum number of kriging neighbours backward, used during iterations
;
WELLDATA
welldata.txt  !Name and location of the welldata file
;
```

Example 2

```
WORKFLOW
0  !estimate_
0  !only_kriging_
1  !use_wells_
0  !use_seismic_
1  !do_iterate_
;
REALIZATIONS
1              !Number of realizations
0              !Header 1 = true 0 = false
C:\Project\    !Output directory and prefix
```

```
;
SIMBOXDEFINITION
0 0 0  !x0, y0, z0
1 1 1 !dx, dy, dz
300 300 1 !nx, ny, nz
;
TI
C:\TrainingImages\channelTI.dat
250 250 1                        !Size x y z direction
;
DIRECTION
!0    !One way simulation
!1    !Back and forth simulation, top to bottom
!2    !Random
3     !Main path is one way, subpaths vary between NWTOP and SETOP
;
FACIES
2       !number of facies
;
CONDITIONING
1     !variogram type !0 = spherical (2D), 1 = empiric (2D and 3D)
30    !variogram range Rx (empiric), variogram range (spherical)
30    !variogram range Ry (empiric), variogram subrange (spherical)
10    !variogram range Rz (empiric), variogram angle (spherical)
3     !maximum number of kriging neighbours
!1    !maximum number of kriging neighbours forward, used during iterations
!1    !maximum number of kriging neighbours backward, used during iterations
;
WELLDATA
welldata.txt  !Name and location of the welldata file
;
LOCALUPDATE
maskslocalupdate.txt    !Name and location of the local update masks file
;
ITERATIONS
1000  !the number of iterations if do_iterate_ == true
25     !lxmin
25     !lymin
0     !lzmin
50     !lxmax
```

```
50     !lymax
50     !lzmax
!variable box to be used during local update, size (2*lx+1)*(2*ly+1)*(2*lz+1)
!only cells listed in input file set in LOCALUPDATE will actually be updated
;
```