The basic algorithm

Snesim stands for “single normal equation simulation” and is an algorithm for simulating the geology of a reservoir. Snesim is a pixel based method that considers multipoint statistics, and combined with its sequential nature it efficiently generates complex realizations such as sinuous channels, incised valleys etc. The idea behind Snesim is very simple, each pixel node is simulated sequentially in a random order. Facies is drawn with conditional probabilities that are frequencies extracted directly from a given training image which reflects a prior knowledge of the reservoir (Srivastava, 1995).

Since it requires a high amount of CPU time to scan the training image for replicates at every node in the simulation path, the concept of a search tree is adopted (Strebelle, 2002). Using a search tree, the training image is scanned once prior to the simulation and all registered events are stored in the tree nodes. This, however, requires a reduction of the amount of conditioning nodes, and only a subset of these nodes can be used. The subset is called a template and is analogous, but not identical, to a neighborhood in a Markov random field sense. See Figure 1 for an example of a square template indicated by the red color. The black and white nodes represent the sand and background facies respectively and the grey nodes are not yet sampled. We note that the use of templates is statistically incorrect since non-sampled nodes within the template depend on nodes outside of the template.

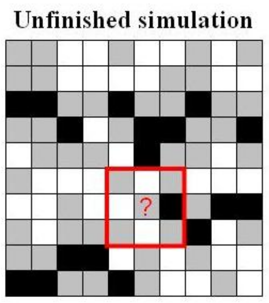

Figure 1: Example of a square template indicated by the red color.

The size of the templates is proportional to the RAM and the CPU time required for the simulations. However, small templates are not capable of capturing the large scale variations, and to compensate for this the simulations are performed on different scales. This is called multiple scale simulation and works as follows; the algorithm proceed first by simulating the coarsest grid *g*, which constitutes of every 2^{g-1}-th node in each direction of the simulation grid, and transfer the simulated values to the next finer grid as conditioning data. This succession continues until the simulation of the finest grid is completed.

The restrictions of Snesim lies in the finiteness of the training image, and all possible configurations of the template are therefore not found in the image. The algorithm cannot detect future inconsistencies since the simulation is performed sequentially, and consequently, conflicts with the training image occur. Snesim solves these conflicts by dropping nodes, which means that the template is reduced until the event of the remaining nodes is recognized. The nodes most distant from the center node are the first to be dropped.

Modification of the algorithm

With node dropping the conflicts are only solved temporarily. Often, the conflicts result in artifacts that are not present in any realistic reservoir. We have made some modifications of the Snesim algorithm by replacing some of the node dropping with node deletion, i.e. we delete conflicting node values and resample them instead of ignoring them. In this way the conflicts that result in artifacts can be solved permanently. We have investigated several possibilities for which nodes to delete and details are given in Table 1 below.

Deletion strategy	**Description**
Delete inwards	Delete from furthest away to nearest node until there is no conflict
Delete outwards	Delete from nearest to furthest away node until there is no conflict
Delete inwards/outwards	Delete either inwards or outwards determined by the least number of nodes that must be deleted
Delete all nodes	Delete all nodes in the template

Table 1: A description of different deletion strategies

Just to illustrate the node deletion we have displayed a template in Figure 2(a), where the nodes within the template are enumerated from nearest to furthest away. In (b) the nodes have been given values, where grey are the non-sampled nodes. If we, for instance, use the outwards deletion strategy and 8 nodes must be deleted, the nodes marked by the red crosses get deleted. Note that in node dropping only the strategy of dropping inwards is used, i.e. the nodes are dropped from furthest away to nearest.

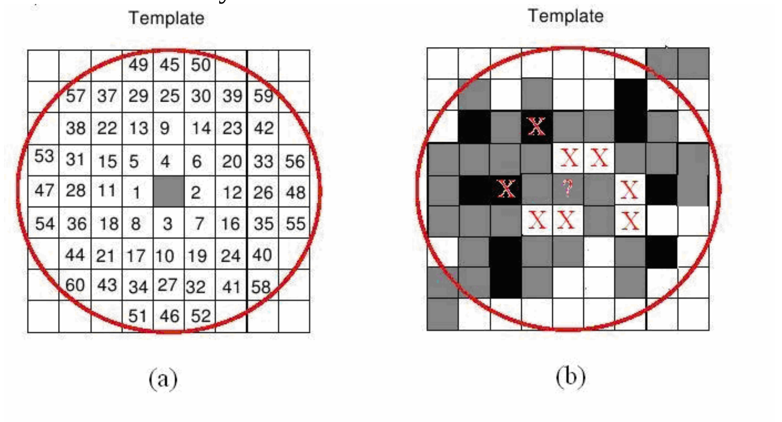

Figure 2: Illustration of node deletion. To the right the nodes within the template are enumerated from nearest to furthest way. The red crosses in the left image are the 8 nearest informed nodes that are deleted.

As mentioned before, the training image represents only a small set of the possible template events, and does not give us any information about the remaining events that can have both high and low probabilities of occurrence. With node dropping, these remaining events are all allowed to occur and the algorithm is not able to distinguish between the various events and their probability of occurrence. In the other extreme, by using node deletion for every conflict none of these remaining events are allowed to occur in the realizations. Thus, the aim is to construct an algorithm where we are able to distinguish between the high and low probabilities of the remaining events, and allow the high probability events to occur. We introduce a tuning parameter, $I_{\max}$, which is a threshold value that classifies the conflicts in two classes; serious conflicts and non-serious conflicts. If the number of sampled nodes is less than $I_{\max}$ and there is a conflict with the training image, the conflict is classified as serious. So, if the conflict is classified as serious the algorithm replaces node dropping with node deletion.

The algorithm becomes iterative once we start deleting nodes, and this raises the question of convergence. Obtaining convergence may be a problem when nodes are deleted and no accepted events can be found, which occurs when $I_{\max}$ is set too low. Thus, this parameter is crucial for the algorithms speed and convergence, and should be adjusted according to the size of the training image and template, and the range of dependency within the training image.

To sum up the modified Snesim algorithm, it works as follows; at each grid level *g* the nodes are visited in a random order, and if there is a conflict which is classified as serious, the conflicting nodes are deleted according to the selected deletion strategy. Otherwise, the nodes are dropped as usual. When all nodes are visited the set of deleted nodes are revisited in a random order. This process is repeated until all nodes on that grid level are informed.

Results

The training image that has been used to test the new Snesim algorithm is displayed in Figure 3 (a) together with two corresponding realizations from the original algorithm in (b) and (c). The number of grid levels that have been used is 3, and the template is circular and of size 60. Note that Snesim could have been run with a larger template to yield better results, but the target here is to improve the results using small templates. For practical problems template size will always be a problem. Both the training image and simulation grids are of size 250×250.

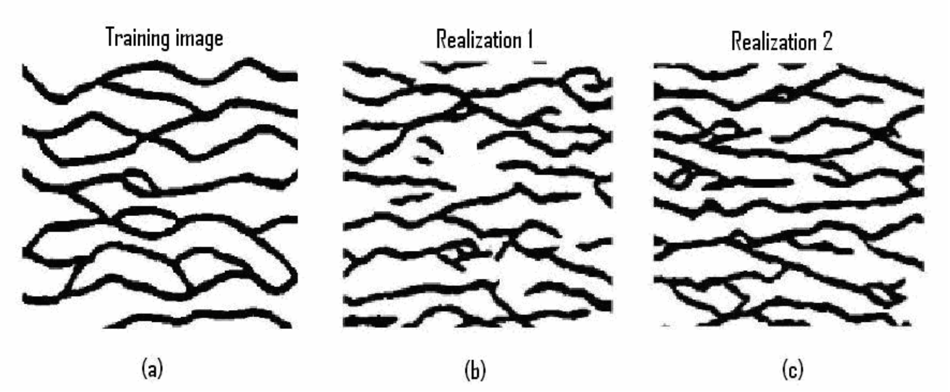

Figure 3: The training image together with two realizations using the original Snesim algorithm.

The realizations exhibit many loose end channels which are not present in the training image. These loose ends are a direct result of conflicts that have been neglected by just dropping nodes. The algorithm allows the template to shrink as much as needed making it possible for these artifacts to occur, and because every simulated node is considered a hard datum it cannot connect two such loose ends once they have been isolated by the other facies type.

We now turn to the modified Snesim algorithm. The parameter $I_{\max}$ is set equal to 30 for the coarsest scale, 40 for the 2nd coarsest scale, and 50 for the finest scale. We increase this parameter value as the grid level becomes finer since we get more sample from the training image at finer scale. The inwards/outwards deletion strategy gave the best simulation results, and realizations from the first and final iteration at each grid level are visualized in Figure 4 (a) – (f). The red nodes represent the nodes that have been deleted and are currently uninformed. At each grid level, the algorithm iterates until all nodes are informed. This strategy was able to connect two loose end channels that were sufficiently close to each other in distance, and it did not erase all loose ends, which was the problem with some of the other strategies

Details regarding simulation time for the modified Snesim are provided by the number of iterations that was required and the number of deleted nodes in each of the iterations. This is given by the plot in Figure 5, which shows the number of deleted nodes versus iteration number at each grid level represented by the colors green, red and blue for the coarsest to the finest grid level, respectively. After approximately 12 iterations, the simulations on each of the grid levels have converged. This is quite fast, and only 23 % of the total 62 500 number of nodes is re-sampled. This is an insignificant increase in CPU time. It is also interesting to see that the finest grid level required the fewest number of iterations, even though this level contains the highest number of nodes.

Thus, iterating using node deletion seems to be a very efficient way of reducing artifacts and achieve at better resemblance with the training image.

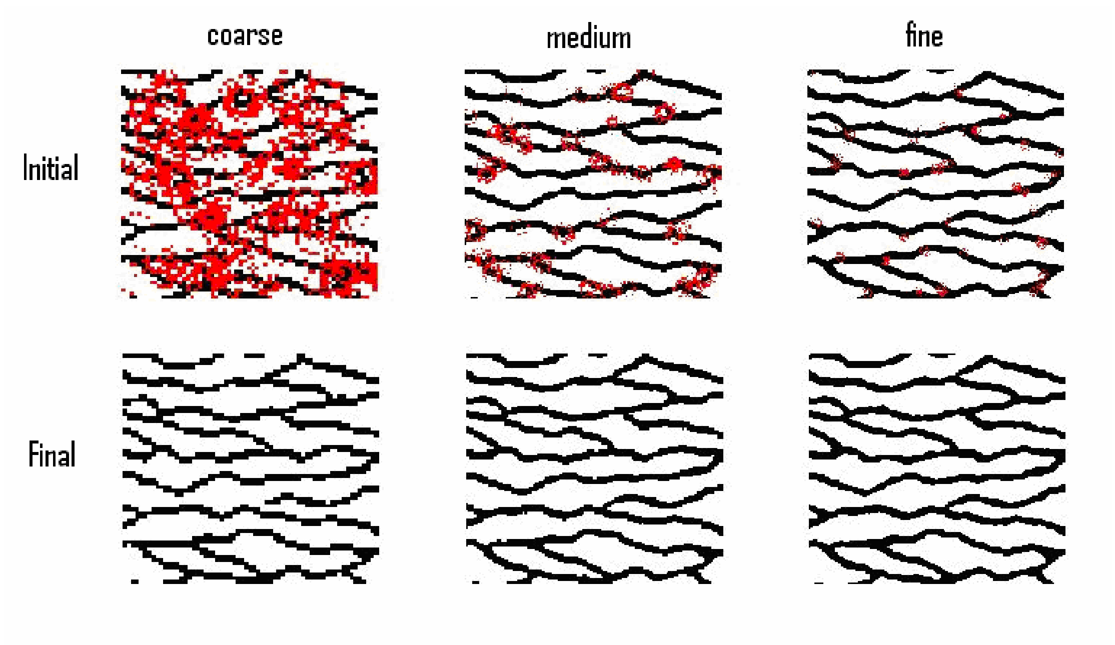

Figure 4: Simulation results using the inwards/outwards deletion strategy. The initial and final iteration at each grid level are visualized.

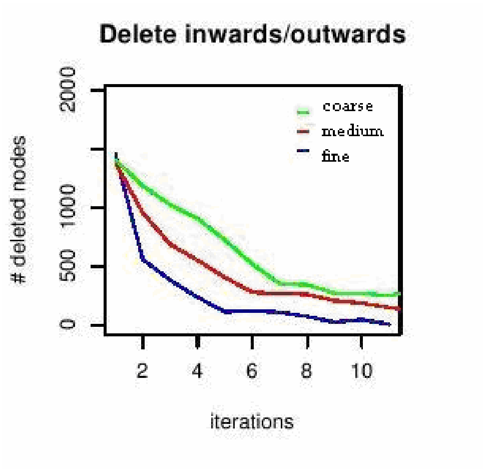

Figure 5: The number of deleted nodes versus iteration number for the inwards/outwards deletion strategy.

References

Srivastava, M.(1995) An overview of stochastic methods for reservoir characterization, in Yarus, J., and Chambers, R., eds., Stochastic modeling and geostatistics: principles, methods, and case studies, v.3: AAPG Computer Applications in Geology, p. 3-16.

Strebelle, S.(2002) Conditional simulation of complex geological structures using multiple-point statistics. Mathematical Geology, v.34, p.1-22.