Guidelines to Perform Multiple-Point Statistical Simulations with the Direct Sampling Algorithm

Eef Meerschman[1], Guillaume Pirot[2], Gregoire Mariethoz[3], Julien Straubhaar[4], Marc Van Meirvenne[5] and Philippe Renard[6]

Abstract The Direct Sampling (DS) algorithm is a recent multiple-point statistical simulation technique. It directly samples the training image (TI) during simulation by calculating distances between TI patterns and a given data event. Omitting the prior storage of all TI patterns in a catalogue, enables to simulate categorical, continuous and multivariate variables. To benefit from the wide spectrum of potential applications of DS, the user needs to understand the role and significance of the input parameters. Therefore, we listed the most important DS parameters and performed a systematic study to assess their significance. Two TIs were used: a categorical image of ice-wedge polygons and a continuous image of a thin marble slice. All DS applications require the definition of three input parameters: the number of neighbors *n*, the acceptance threshold *t* and the fraction of TI to scan *f*. We investigated the influence of these parameters on CPU time and simulation quality for the 3D parameter space. It was found that decreasing *f* offers substantial computational gains without an important degradation of the simulation quality, whereas adjusting *t* and *n* to decrease CPU time comes at the expense of lower simulation quality. We also illustrated the noise removal potential of post-processing and the possibility to simulate bivariate fields and to honor conditioning data. For each case, the relevance of the required input parameters was evaluated. We provided a comprehensive guide to performing multiple-points statistical simulations with the DS algorithm and recommendations on how to set the input parameters appropriately.

[1] Research Group Soil Spatial Inventory Techniques, Department of Soil Management, Faculty of Bioscience Engineering, Ghent University, Coupure 653, 9000 Gent, Belgium, eef.meerschman@ugent.be

[2] Centre of Hydrogeology and Geothermics, University of Neuchâtel, Rue Emile Argand 11, CH-2000 Neuchâtel, Switzerland, guillaume.pirot@unine.ch

[3] National Centre for Groundwater Research and Training, School of Civil and Environmental Engineering, The University of New South Wales, Sydney, Australia, gregoire.mariethoz@minds.ch

[4] Centre of Hydrogeology and Geothermics, University of Neuchâtel, Rue Emile Argand 11, CH-2000 Neuchâtel, Switzerland, julien.straubhaar@unine.ch

[5] Research Group Soil Spatial Inventory Techniques, Department of Soil Management, Faculty of Bioscience Engineering, Ghent University, Coupure 653, 9000 Gent, Belgium, marc.vanmeirvenne@ugent.be

[6] Centre of Hydrogeology and Geothermics, University of Neuchâtel, Rue Emile Argand 11, CH-2000 Neuchâtel, Switzerland, philippe.renard@unine.ch

Introduction

Most multiple-point statistical simulation algorithms first build a conditional cumulative distribution function F conditioned to a local data event $\mathbf{d}_n$ for each unknown location of the simulation grid $\mathbf{x}$. The Direct Sampling (DS) algorithm skips the explicit modeling of this ccdf. For each $\mathbf{x}$, the training image (TI) is randomly scanned. As soon as a TI pattern is found that matches $\mathbf{d}_n$ exactly or as soon as the distance between a TI pattern and $\mathbf{d}_n$ is lower than a user-defined threshold, the value at the central node of this TI pattern is pasted to $\mathbf{x}$. Because of this strategy, DS allows to simulate both categorical and continuous variables, and to handle multivariate cases. For each variable type, one only has to select the appropriate dissimilarity distance [1].

To run DS, the user needs to define several input parameters. Since DS is a promising, but very recent technique, we performed a sensitivity analysis on the most important input parameters aiming to encourage (potential) users to benefit from the wide spectrum of applications of DS.

Main input parameters: *n*, *t* and *f*

Parameter n is the maximum number of known grid nodes $\mathbf{x}_i$ (including conditioning data and already simulated grid nodes) within a defined search area that will form $\mathbf{d}_n$. Hence, the area covered by $\mathbf{d}_n$ decreases when the number of already simulated grid nodes increases, ensuring that structures of all sizes are present in the simulation. Parameter t is the acceptance threshold for a TI pattern to be selected. Since all distance types are standardized between 0 and 1, t will also be in this range. Parameter f is the maximum fraction of the TI that is scanned for each $\mathbf{x}$ and also ranges between 0 and 1. If no TI pattern is found that has a distance lower than t, the TI pattern with the lowest distance is selected [1].

For 1200 different combinations of n, t and f we simulated 10 unconditional realizations for each TI. Each parameter strongly influenced CPU time: simulations with a large number of neighbors n, a low acceptance threshold t and a large fraction of TI to scan f required a long simulation time. Consequently, the combined effect of relaxing all three parameters only slightly, can drop off CPU time. For instance, generating one unconditional simulation for the categorical case with parameters $t = 0.05$, $f = 0.5$ and $n = 50$ took 163 s. Relaxing t to 0.1 only took 44 s, relaxing all three parameters to $t = 0.1$, $f = 0.3$ and $n = 30$ only took 13 s.

Figure 1 illustrates how n, t and f influenced the simulation quality. The connectivity error is a measure for the dissimilarity between the connectivity function of the TI and the unconditional simulations summed over the different categories. Whereas n and t strongly influenced the simulation quality, the effect

of f was much smaller. Scanning a smaller part of the TI hardly resulted in a quality decrease. This could also be concluded for the continuous TI.

Generally we recommend to choose $n \geq 30$ and $t \leq 0.2$ for categorical TIs and $t \leq 0.1$ for continuous TIs. This choice depends on the affordable CPU time and the required simulation quality. Furthermore, for large n and small t the user should check if there is still sufficient variability between the simulations, because one risks to pick the same best matching nodes in each realization ('patching'). A good strategy to reduce both CPU time and the risk of patching is setting $f < 1$, thus scanning a different fraction of the TI for each **x** [2].

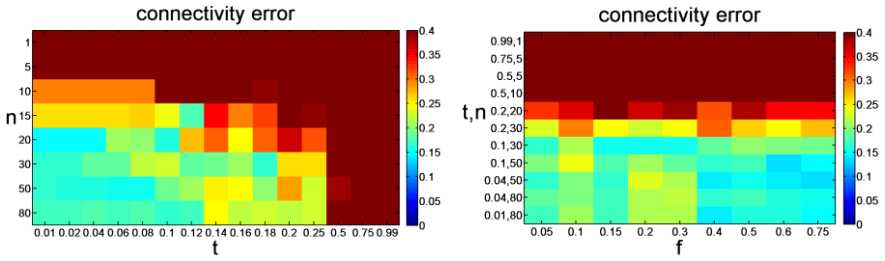

Figure 1: Influence of t and n (for $f = 0.5$) (left) and f (right) on the simulation quality for the categorical TI.

Post-processing for noise removal

DS includes a post-processing option targeting the removal of isolated pixels (noise) in the simulated images. After having simulated all the unknown grid nodes, it is possible to resimulate each node with an entirely informed neighborhood. Two post-processing parameters need to be defined: the number of post-processing steps p and the post-processing factor p_f. The latter is the factor by which f and n are divided aiming to save CPU time in the additional post-processing steps [1]. Figure 2 illustrates the noise removal effect of post-processing for the categorical ice-wedge TI.

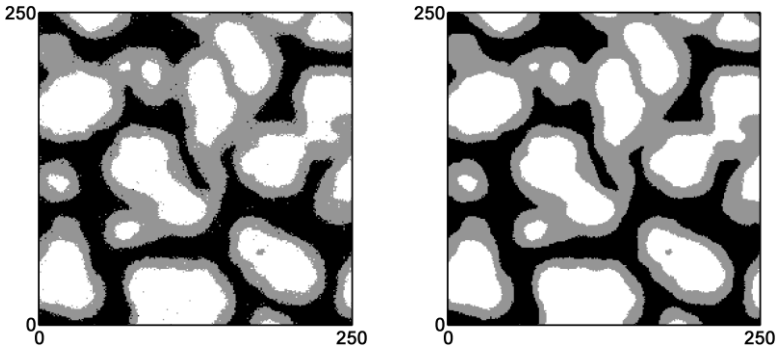

Figure 2: Unconditional simulation with $n = 50$, $t = 0.1$ and $f = 0.5$ without post-processing (left) and with one post-processing step ($p = 1$ and $p_f = 1$) (right).

By running different test cases, we found that adding an extra post-processing step gave the best results for categorical variables and intermediate t values (between 0.1 and 0.2). Next to a decrease in the level of noise, post-processing allowed for a significant reduction in CPU time. With $t = 0.1$ and one post-processing step, one obtained in 58 s realizations similar to when using $t = 0.05$ without post-processing, which took 163 s. Although the post-processing option also reduced the level of noise in the continuous case, its effect was less substantial than for the categorical case and the CPU cost was much higher. For both TIs the simulation quality was insensitive to parameters p and p_f [2].

Hence, it can be considered as good practice to always add one post-processing step ($p = 1$) with $p_f = 1$ when simulating categorical variables. If the simulations still contain (too much) noise, parameters t and n should be adapted.

Multivariate simulations

Simulating multivariate images is a very new and promising technique first offered by the DS algorithm. To run multivariate simulations the user needs to construct a multivariate TI which not only represents the expected spatial structure of each variable, but also the expected relationship between the m variables. The selection of an appropriate multivariate TI pattern for each $\mathbf{x}$ is based on a weighted averaged of the m selected distance measures. These weights can be chosen by the user [1]. Bivariate simulations with one categorical and one continuous variable showed that a larger weight given to the continuous variable strongly improved its simulation quality [2].

Data conditioning

DS always honors available conditioning data since these are located to the closest grid node prior to simulation. It is possible to give these conditioning grid nodes a larger weight than the already simulated nodes when calculating the distance between $\mathbf{d}_n$ and the scanned TI patterns [1]. We found that making use of this feature strongly improved the coherence of the simulated spatial patterns with the conditioning data and prevented them to appear as isolated pixels. When the expected uncertainty of the conditioning data is low, giving a weight to the conditioning grid nodes that is up to five times larger improves the pattern consistency [2].

Conclusion

We reported a comprehensive sensitivity analysis for the DS algorithm that will help users to benefit more efficiently from the potential of DS. We concluded that it is advised to set $n \geq 30$ and $t \leq 0.2$ for categorical simulations and $t \leq 0.1$ for continuous simulations. The smaller t and the larger n, the better the simulation quality is and the lower the level of noise, but the higher the required CPU time. For small t and large n the user should check if there is still sufficient variability between the simulations. Setting $f < 1$ is a good strategy to reduce both CPU time and the risk of patching. Especially for categorical simulations, we recommend to always add one post-processing step for noise removal. When simulating bivariate images, the weights given to each variable clearly affect simulation quality and when conditioning data are available, it is interesting to put the weights given to the conditioning data higher than the weights given to the already simulated nodes.

Bibliography

1. Mariethoz, G., Renard, P., Straubhaar, J., 2010. The Direct Sampling method to perform multiple-point geostatistical simulations. Water Resources Research 46, W11536.
2. Meerschman, E., Pirot, G., Mariethoz, G., Straubhaar, J., Van Meirvenne, M., Renard, P. A practical guide to performing multiple-point statistical simulations with the Direct Sampling algorithm. Submitted for publication in Computers & Geosciences.