Improvements in Continuous Variable Simulation with Multiple Point Statistics

Jeff B. Boisvert

A modified version of Mariethoz et al's (2010) algorithm for simulating continuous variables using multiple point statistics is presented. The features of the training images are well reproduced in the resulting simulations; however, the user is required to define a number of tolerances. The modification presented in this work focuses on improving the search for a pattern during simulation. In general, patterns in the TI are found quicker and are more similar to the desired template.

Introduction

Multiple point statistics (MPS) is commonly used to generate complex categorical models and many algorithms are available for implementing MPS with categorical variables. Continuous variables are more difficult to simulate as the multiple point statistic cannot be summarized by a simple count of unique patterns. A promising MPS algorithm for continuous variables is presented by Mariethoz et al (2010). The idea is to search for surrounding data in the realization, find the nearest *n* simulated nodes, scan the training image (TI) directly and calculate the difference between the pattern in the simulation and the pattern in the TI. A tolerance for accepting a sufficiently similar pattern in the TI is defined by the user; when a pattern is found with a tolerance less than the specified tolerance, the value in the TI is used as the simulated value in the realization. Mariethoz et al (2010) version of the algorithm scans the TI randomly until a sufficiently similar pattern is located. The proposed algorithm in this work uses a sorted TI in order to more quickly locate similar templates in the TI. This has a twofold improvement 1) relevant patterns are found quicker, reducing the simulation time and 2) patterns with a lower error can be found, reducing the overall error as measured by the average pattern dissimilarity in a realization.

The methodology section presents the algorithm. Next, the FORTRAN program MPS_CONT.exe is described with various options. Finally a 2D example demonstrates the technique. Note that the algorithm is fully compatible with 3D TI's and realizations; however, for this work, a 2D TI will be sufficient for demonstration purposes. The algorithm scales linearly to 3D with the number of patterns in the TI and the user defined number of nearest nodes to retain for simulation. Note that an additional benefit of this methodology is that an MPS template is not required. The user specifies the number of nearest samples to use and is not limited by a fixed template.

Note that the generation of a suitable 2D or 3D TI is not discussed here. It is the users responsibility to generate the TI; however, some guidance for using blast hole data as a TI is provided in (Paper 304 in this report).

Methodology

The general paradigm of MPS simulation is to find the appropriate local distribution of a variable of interest conditional to previously simulated values. Typically, a pattern or MPS template is defined; all previously simulated values falling in this template are used as conditioning data (similar to a search template with traditional Gaussian based techniques). All locations in the TI that match this pattern are found and the local distribution is defined by the TI values corresponding to the central template node. A value is drawn at random from this distribution and included as a previously simulated value for future locations. This is repeated for all locations in the model and a realization is generated.

The difficultly is in finding similar patterns in the TI. This is made easier if the variable can only contain a finite number of discrete values, such as with facies codes and rock types; in this case, efficient search trees can be used to store all patterns and local distributions can be quickly extracted from the tree. If the variable of interest is continuous, a search tree cannot be used in the same way. Mariethoz et al (2010) propose the following MPS algorithm that is appropriate for continuous variables (Figure 1):

Step A) Generate an appropriate TI.

Step B) Search for *n* nearest neighbors in the realization. The technique does not require a fixed template, similar to Gaussian based techniques, it implements a user defined number of nearest previously simulated nodes. Typically this number is selected to be less than 60 for computational efficiency.

Step C) Using the *n* nearest neighbors as the template, directly scan the TI for similar patterns. A starting location is selected randomly within the TI and the scan progresses through the image in order.

Step D) Calculate the difference between the data found in the realization and the pattern in the TI. This is done using Eqn 1.

Step E) When a pattern in the TI is found that is less than a user defined tolerance, the value in the central node is used as the simulated value for the realization.

Step F) Repeat for all locations in the model.

Any criterion could be used to determine pattern similarity. In this work, we will consider the mean squared difference between nodes in the TI and nodes in the realization (Figure 1). The differences are weighted based on the distance of each node to the central node as in Mariethoz et al (2010) (Eq. 1).

$$d\{\mathbf{d}_n(\mathbf{x}), \mathbf{d}_n(\mathbf{y})\} = \sqrt{\sum_{i=1}^{n} \alpha_i [Z(\mathbf{x}_i) - Z(\mathbf{y}_i)]^2} \quad (1)$$

where α_i is the inverse distance to the central node.

The proposed improvement to this methodology is in the random selection of potential TI patterns. Rather than randomly selecting a location in the TI to begin scanning, the TI is first sorted by value, and the nearest conditioning data in the realization (hereafter referred to as the reference node) is used to find better starting patterns to scan; consider the following TI (Figure 1). The TI is sorted and the simulation begins by finding the TI patterns that are most similar to the reference node. This is straightforward when the values in the TI have been sorted. The TI is then scanned in order of difference with the reference node. This requires very little additional memory; the TI is stored in sorted order with a pointer for each value pointing to the range of locations to scan in the TI. The user can provide a threshold on the reference node or it can be taken as a fraction of the original tolerance.

Consider the following example with the TI borrowed from Zeng et al (2006), one realization using Mariethoz et al (2010) and one realization with the 1D search method (Figure 3). Two metrics are used to assess the models 1) the average pattern difference, recall the mean squared error is used to determine patter similarity, the average pattern difference is the average over all simulated locations 2) the CPU time. Figure 4 shows that the proposed 1D search method generates realizations in less time with less error. The various lines on the plots indicate different threshold parameters for the algorithms, which are identical for both implementations. Note that the average pattern difference is important, three realizations with different average pattern differences are shown in Figure 5. As the pattern difference is reduced, the realizations better match the features in the TI.

Program

The parameters for MPS_MDS are explained below. Note that only one realization is currently generated with this program. This will be modified in future releases. Note that the intention is to use the parallel program (CCG paper #409 in report 12, 2010) in order to generate multiple realizations in parallel.

```
                 Parameters for MPS_MDS
                 **********************

START OF PARAMETERS:
01    data.dat        -file with data
02    1 2 0 3         -columns for X,Y,Z,var
03    -998   1.0e21   -trimming limits
04    0               -debugging level: 0,1,2,3
05    debug1.dbg      -file for debugging output
06    debug2.dbg      -debug file
07    MPS_real_1.out  -file for simulation output
08    200 0.5 1       -nx,xmn,xsiz
09    200 0.5 1       -ny,ymn,ysiz
10    5  0.5 1        -nz,zmn,zsiz
11    1 3             -use a multi grid (1=yes, 0=no), number of grids
12    2 20            -min, max data
13    100             -maximum search radii
14    -1              -maximum number of dimensions to use in search
15    3               -MPS option, 1=kd tree (does not work well), 0=Mariethoz implementation, 2=exhaustive, 3=1D search method
16    10              -nclusters for MDS scaling, option 1
17    ../TI_3D.out    -training image
18    900 900 5       -size of the TI
19    65421           -random number seed
20    200  100000     -pattern difference threshold, max patterns to check, used in most methods
21    1               -for option 3, need the minimum difference for building the 1D search array
22    100             -only for option 3, option to consider a minimum number of patters at each point, 100 works well, 0=ignore
23    0               -only for option 3, if the TI has missing values, have to restandardize, set option to 1
```

Line 1 - data file
Line 2 - columns for the coordinates and value
Line 3 - trimming limits on the variable to be simulated
Line 4 - debugging level (not used currently)
Line 5/6 – some debugging information, different depending on the MPS option selected
Line 7 – simulation file output
Line 8/9/10 – grid definition
Line 11 – multi grid approach, similar to SGSIM
Line 12 – number of previously simulated values to use, data are treated as identical to previously simulated values
Line 13 – isotropic search radii for nearest neighbors
Line 14 – for MPS option #1, this is the number of dimensions to use, ignored for other options
Line 15 – MPS option to consider, Option 0 is to use Mariethoz's implementation (as described above), Option 2 searches the entire TI for the best pattern (very slow, but the ideal simulation) Option 3 considers the sorting and 1D search using the reference node approach described above. Option 1 uses a multidimensional scaling embedding of the TI patterns and then a search in high-dimensional space. This option does not work well and is not recommended.
Line 16 – number of clusters, only used for option 1, ignore
Line 17 – file with the TI
Line 18 – size of the TI, the blocks are assumed to have the same dimension as the realization (i.e. the TI is at the same scale as the realization)
Line 19 – random number
Line 20 – user defined tolerance for pattern similarity. This is not the average pattern similarity; it is the **sum of the squared differences**. Using more data points will result in a larger squared difference. Moreover, if the maximum number of patters to check is reached, the TI pattern with the minimum difference is selected even though the threshold has not been reached
Line 21 – when building the sorted lookup tree for determining the patterns to check, the tolerance on the reference node is used to determine which patterns will be checked. This is highlighted in Figure 1 where a tolerance of 0.1 is used. The value required here is the absolute difference, not the squared difference.
Line 22 – if the first pattern that is check is less than the tolerance, it may be beneficial to test a few more patterns. The minimum number of patterns to check is defined for option 3 (not implemented in other techniques).
Line 23 – it could be the case that the TI is not exhaustive. In this case certain pattern points are dropped and the sum of the squared difference (Line 20) is modified to consider dropping nodes in the pattern.

Conclusion

The proposed method shows promise for improving simulation results when considering MPS simulation with continuous variables. The logical extension is to generate a 2D search where two points are used as reference nodes than just the nearest reference node. This would significantly improve the algorithm in terms of CPU speed and average pattern error; however, the memory requirements of a 2D lookup table, rather than the 1D sort used here, would be large.

References

Mariethoz, Renard and Straubhaar (2010) The Direct Sampling method to perform multiple-point geostatistical simulations. Water Resources Research. Vol 46. 14 p.

Zhang, Switzer, and Journel (2006) Filter-based classification of training image patterns for spatial simulation, Math. Geol. 38(1), 63–80

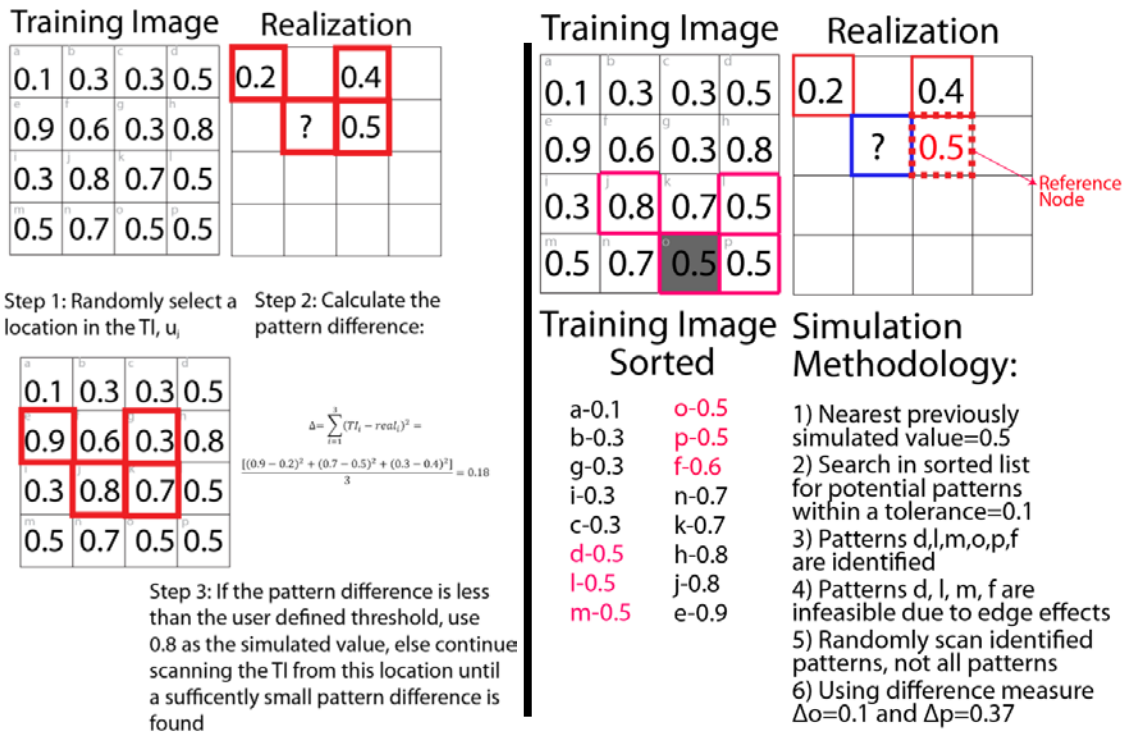

Figure 1: Left: Methodology for Mariethoz's implementation. Right: Methodology for 1D search implementation.

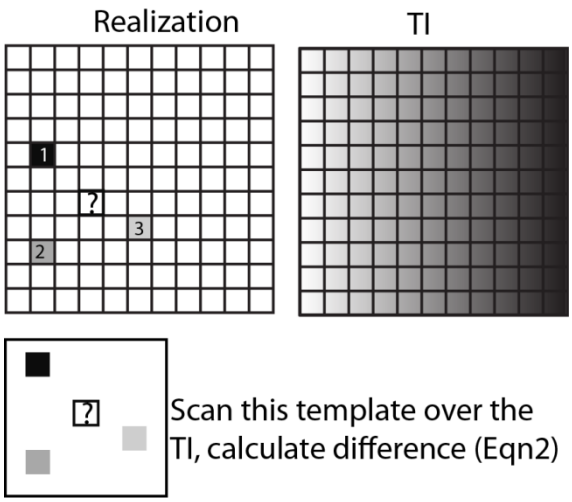

Figure 2: Conditioning data to use in MPS. No template is specified, the user provides the number of nearest conditioning data and previously simulated nodes to use.

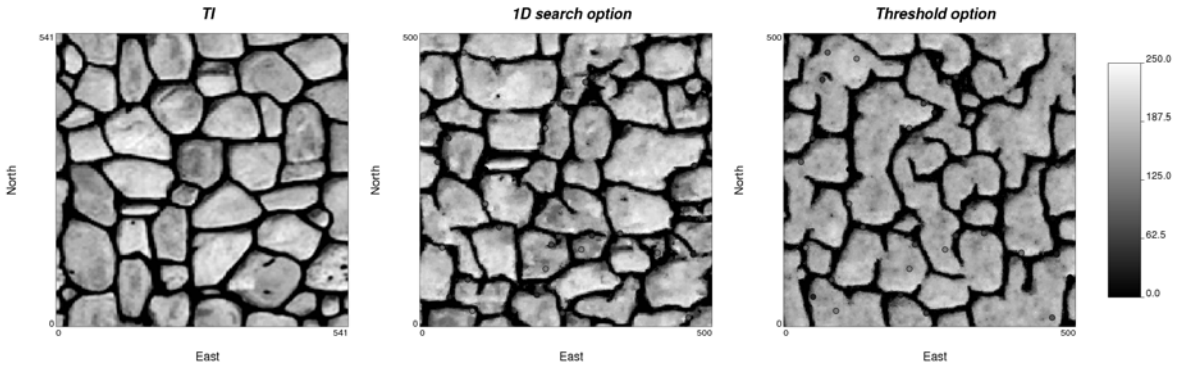

Figure 3: Left: TI. Middle: realization with data used, option #3 the 1D search. Right: Same tolerance parameters, but using option #0 the threshold implementation. Average error for middle realization is 1961 average error for right realization is 2831. Time for middle realization is 225s and time for right realization is 333s.

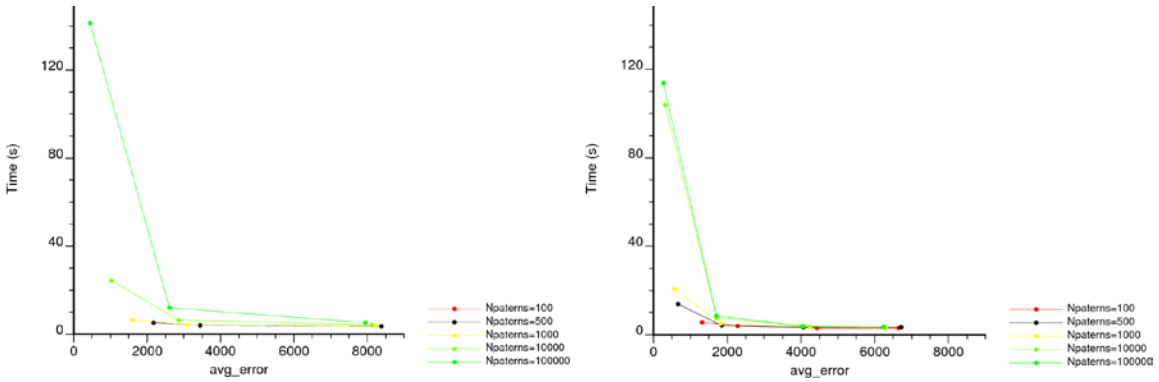

Figure 4: Left: Times and errors using option #0. Right: Times and errors using the 1D search option #3. Lines represent different total number of patterns searched in the TI. Points along the line are generated by lowering the pattern difference threshold.

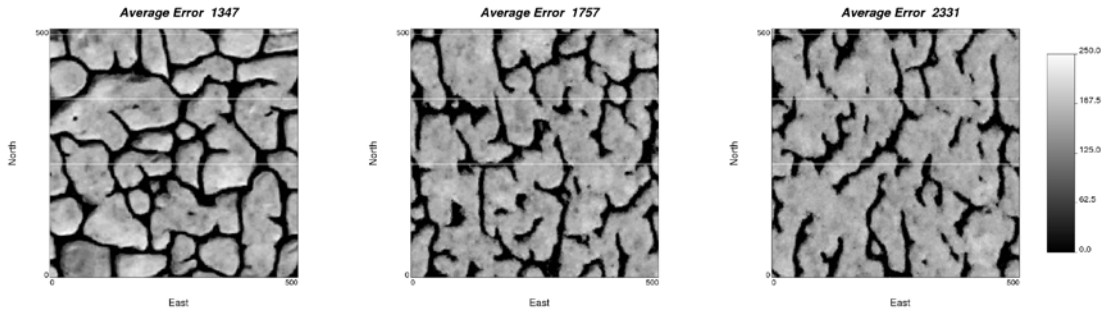

Figure 5: Three examples of realizations with different average pattern errors.