FORTRAN Programs for Calculating Connectivity of 3-D Numerical Models and for Ranking Multiple Realizations

C. V. Deutsch (cdeutsch@civil.ualberta.ca),
Department of Civil & Environmental Engineering, University of Alberta

Abstract

Given a 3-D numerical geological model it is often useful to determine the geological modeling cells that are connected in 3-D. These connected 3-D bodies are referred to as geo-objects*; a program* `geo_obj` *is presented to calculate the geo-objects from 3-D lithofacies, porosity, and permeability models. Relevant summary statistics such as the number of geo-objects, their sizes, and their tortuosities are reported.*

The connected volumes may be used for ranking geostatistical realizations, e.g., establishing realizations with low*,* median*, and* high *connectivity. A program* `rank_obj` *is presented for this purpose. It may also useful to rank realizations according to the connectivity to one particular location (e.g., a well location) or between two locations (e.g., injector-producer pair); programs* `rank_loc` *and* `rank2loc` *are presented.*

The concept of 3-D connectivity and the topology of 3-D Cartesian grids are well understood; however, the programs presented here will be useful to anyone dealing with 3-D numerical geological models.

KEY WORDS: geo-objects, ranking, geostatistical realizations, indicator simulation

Introduction

Consider 3-D Cartesian arrays of lithofacies f, porosity ϕ, and permeability k. A null lithofacies type ($f = 0$) is used to identify those regions of the 3-D model outside the volume of interest. For example, certain cells at the top of the model may have been truncated by more recent geological events, faults may compartmentalize the volume, or certain regions may be too far from data to be considered. The data arrays:

$$f(\mathbf{u}_{i,j,k}), \phi(\mathbf{u}_{i,j,k}), k(\mathbf{u}_{i,j,k}): \quad i = 1, \ldots, nx, \;\; j = 1, \ldots, ny, \;\; k = 1, \ldots, nz$$

where $\mathbf{u}_{i,j,k}$ is the location of block i, j, k. In this Cartesian grid system, location $\mathbf{u}_{i,j,k}$ is *face*-connected to six other cells, *edge*-connected to an additional twelve cells, and *corner-point*-connected to a further eight cells, see Figure 1. Most often, we consider *face*-connected cells because of our interest in connected volume analysis; however, the source code can additionally handle *edge* and/or *corner-point*-connections.

We will define an indicator variable at each cell according to whether it meets some minimum criteria for connectivity. Cells are considered *net* if they meet the criteria and *non-net* if they do not. The following scheme is used to code each cell:

- initialize cell i, j, k to *net*
- if $f_{i,j,k}$ does not belong to a *net* lithofacies type then reset cell i, j, k to *non-net*
- if $\phi_{i,j,k}$ is less than some critical porosity threshold then reset cell i, j, k to *non-net*
- if $k_{i,j,k}$ is less than some critical permeability threshold then reset cell i, j, k to *non-net*

Only one of f, ϕ, k is required to define the *net* indicator. All subsequent analysis will use this net indicator. It is recommended that sensitivity studies be carried out to be ensure that the conclusions drawn from the connectivity analysis are robust with respect to the porosity and permeability thresholds chosen.

An alternative to using a 3-D geological model would be to directly use seismic data, see Gutteridge and Gawith, 1996. The *net* indicator would be defined by a threshold applied to the seismic data. Any other exhaustively known gridded data attribute could be used.

Algorithm for Geo-Object Calculation

A full discussion of the various methods to establish connected 3-D volumes is outside the scope of this paper, see Mehlhorn or Preparata for more details. A simple approach is presented and coded. The improvements in CPU speed that may be obtained with more complex methods would be offset by more obscure source code. Easily accessible source code is preferred since, for models with less than 1 million cells, the CPU time of the simple algorithm presented here is not considered excessive.

Figure 2 shows the CPU time (on a SGI Indigo2) versus model size for models between 1 thousand cells and 1 million cells; the CPU time for the 1 million cell model is just under 25 minutes. These CPU times are for 3-D `sisim` models with a 0.5 fraction of *net* cells. The CPU times do not change by more than 5% when the fraction of net cells is significantly decreased or increased. Also, these CPU times are for face-connections only; adding edge and corner connections increases the CPU time by 12-15%.

The algorithm consists of scanning through the 3-D array aggregating those cells that are *connected*. The algorithm proceeds as follows:

1. Establish the binary *net* indicator: each cell i, j, k is *net* or *non-net*
2. Set first-pass number of geo-objects to zero ($n' = 0$) and then scan through the model (arbitrarily choose to loop over X fastest, then Y, then Z) coding each *non-net* cell as a 0 and each *net* cell as (1) the *net* code of an adjacent cell, or (2) increment $n' = n' + 1$ and code the cell as n'
3. Set counter equal to 0 ($nch = 0$). For each Y stack of cells, corresponding to an X and Z index, loop along Y stack checking cell j with $j - 1$. If both are *non-net* cells and have different net codes then increment number to change $nch = nch + 1$ and record $from(nch)$ and $to(nch)$.
4. Make the nch changes corresponding to scan over Y stacks:
   - Reset $from/to$ to ensure that $to(l) < from(l)$ for all $l = 1, \ldots, nch$

- Remove duplicate $from/to$ pairs (could reduce nch)
- Add $from/to$ pair when $from(l) = from(l'); l \neq l'; l, l' \in (1, nch)$, that is, $nch = nch + 1$, $from(nch) = max(to(l), to(l')$, $to(nch) = min(to(l), to(l')$.
- Set common $from$ pairs ($from(l) = from(l'); l \neq l'; l, l' \in (1, nch)$) to lowest common to, that is, $to(l) = to(l') = min(to(l), to(l')$.
- Loop over all cells in model resetting cells with net code equal to $from(l)$ to $to(l), l = 1, \ldots, nch$.

5. Reset number to change equal to 0 ($nch = 0$). For each Z stack of cells, corresponding to an X and Y index, loop along Z stack checking cell k with $k-1$. If both are *non-net* cells and have different net codes then increment number to change $nch = nch + 1$ and record $from(nch)$ and $to(nch)$.

6. Make the nch changes corresponding to scan over Z stacks (as in step 4). Now, the *net* codes correspond to groupings of face connected geo-objects.

7. If edge connections have been requested, scan through the model (arbitrarily choose to cycle through X then Y then Z). Skip over *non-net* cells. For *net* cells, loop over all 12 connected edge cells:

   - if the adjacent cell is *non-net*, then there is nothing to be done
   - if the adjacent cell is *net*
     - determine which of the two cells has the lowest integer code
     - scan through all cells reseting the largest integer code to the smallest
   - after this step, adjacent *net* cells (face and edge connected) will have the same integer geo-object code.

8. If corner connections have been requested, scan through the model (arbitrarily choose to cycle through X then Y then Z). Skip over *non-net* cells. For *net* cells, loop over all 8 connected corner cells and reset to lowest geo-object code (as in 7). After this step, adjacent *net* cells (face and corner connected) will have the same integer geo-object code.

9. Determine the number of geo-objects $n_{geo_{obj}}$ by the number of unique integer codes (excluding 0) in the model.

10. Calculate the size of each geo-object

11. Sort the geo-objects by size and number them from 1 to $n_{geo_{obj}}$ with 1 corresponding to the largest and $n_{geo_{obj}}$ corresponding to the smallest.

The result is a 3-D integer-coded grid where non-net cells are 0 and net cells are assigned a geo-object number. The geo-object numbers increase from 1 to $n_{geo_{obj}}$ where $n_{geo_{obj}}$ must be less than or equal to one half the number of net cells in the model. The size of each geo-object is also known.

It is often interesting to know the *tortuosity* of each geo-object since two geo-objects with the same volume may have different connectivity characteristics if one is highly connected

and the other is very tortuous with small tenuous connections. One measure of tortuosity is the ratio of surface area to volume; for a fixed volume, the greater the surface area the more tortuous the object. Both volume and surface area are easily calculated from the 3-D geo-object grid. The surface area is established by scanning through all cells in a geo-object and calculating the number of "outside" faces.

A FORTRAN implementation of this algorithm is coded in program `geo_obj`. This program was modeled after the GSLIB programs (Deutsch and Journel, 1992). The input parameters are shown on Figure 3 and documented below:

- **lithmod:** input file with lithofacies model. This is an ASCII file of integer-coded lithofacies types. After a three line header (see example) the lithofacies codes are specified one cell at a time with the X index cycling fastest, then Y, and then Z.
- **poromod:** input file with porosity model. The ASCII input file of porosity also has a three line header and the porosity values with the X index cycling fastest, then Y, and then Z.
- **permmod:** input file with permeability model. The ASCII input file of permeability with a three line header and the permeability values with the X index cycling fastest, then Y, and then Z.
- **outfl:** output file for 3-D specification of geo-object codes. An ASCII file containing the geo-object code for each cell. Non-net cells are coded 0 and the geo-objects are sorted according to size (1=the largest, 2=the second largest, ...). The output grid has the X index cycling fastest, then Y, and then Z.
- **sumfl:** output file for summary statistics of geo-objects. This file contains the number of geo-objects and their sizes.
- **nx, ny, nz,** and **nsim:** the number of cells in the X direction, Y direction, Z direction, and the number of realizations to consider
- **nnet** and **netcode(i),i=1,...,nnet:** the number of net lithofacies codes and those integer codes.
- **porcut:** the minimum porosity threshold to define *net*
- **percut:** the minimum permeability threshold to define *net*

Ranking Realizations with Geo-Object Connectivity

One application of geo-object connectivity is in ranking realizations from *low* to *high* connectivity. Ranking may only be performed according to a scalar, that is, a single number. Often, the character of a 3-D model is not summarized by a scalar. The *average* measures of connectivity considered here, however, provide one way to rank realizations.

`rank_obj` reads in the summary file output by `geo_obj` for multiple realizations and ranks the realizations by three criteria: (1) the cumulative number of connected cells within the first n_1 geo-objects, (2) the fraction of *net* cells within the first n_2 geo-objects, and (3) the

tortuosity of the first n_3 geo-objects. The parameters for `rank_obj` are shown on Figure 4 and documented below:

- **sumfl:** input file of summary statistics of geo-objects from program `geo_obj`. This file contains the number of geo-objects and their sizes for all realizations.
- **outfl:** output file with rank of all realizations by three individual measures and a rank based on the average of the three rank measures.
- **n_1:** number of geo-objects to use for calculating a cumulative volume of connected cells.
- **n_2:** number of geo-objects to use for calculating the cumulative net fraction.
- **n_3:** number of geo-objects to use for establishing an average tortuosity for ranking.

At times we know a particular location of interest within the 3-D volume, e.g., the location of a production well or the location of a contaminant source. In such a case, the realizations could be ranked by the connected volume within some radius of that location. The location could be a single cell or any arbitrary collection of cells relating to a deviated well or a fluid contact. For maximum flexibility, in *rank_loc* the well blocks must be entered explicitly. The ranking is made with the cumulative connected cells to an arbitrary number of well blocks. The parameters for `rank_loc` are shown on Figure 5 and documented below:

- **datafl:** the output file of `geo_obj` with the 3-D specification of geo-object codes.
- **sumfl:** output file for connected cells and ranking measure.
- **nx, ny, nz,** and **nsim:** the number of cells in the X direction, Y direction, Z direction, and the number of realizations to consider
- **xsiz, ysiz,** and **zsiz:** the constant size of all the cells to calculate a total connected volume
- **radius:** the maximum distance from the location(s) of interest to consider a cell still connected, i.e., all cells beyond **radius** of a location of interest are not considered.
- **nloc:** the number of well locations to be considered

  For every well:
- **nblock(iwell):** the number of well blocks for this well.
- **ix(i), iy(i),** and **iz(i):** the well blocks.

There are other times when the process of interest depends on the connected volume between two locations, e.g., between producer-injector pairs of wells. In such cases, the realizations could be ranked by the connected volume between the two locations (and yet within some radius of the line connecting the locations). As before, the assumption in *rank2loc* is that the well locations are specified by some arbitrary number of cells or blocks. The ranking is made with the cumulative connected cells between an arbitrary number of pairs of wells. The parameters for `rank2loc` are shown on Figure 6 and documented below:

- **datafl:** the output file of `geo_obj` with the 3-D specification of geo-object codes.
- **outfl:** output file for connected cells and ranking measure.
- **nx, ny, nz,** and **nsim:** the number of cells in the X direction, Y direction, Z direction, and the number of realizations to consider
- **xsiz, ysiz,** and **zsiz:** the constant size of all the cells to calculate a total connected volume
- **radius:** the maximum departure distance from the path that connects the two locations by the shortest distance. All cells beyond **radius** of the line connecting the two locations are not considered.
- **nloc:** the number of well pairs to be considered

  For every well pair:

- **nblock1(iwell):** the number of well blocks for the first well.
- **ix1(i), iy1(i),** and **iz1(i):** the well blocks for well 1.
- **nblock2(iwell):** the number of well blocks for the second well.
- **ix2(i), iy2(i),** and **iz2(i):** the well blocks for well 2.

A Small Example

Consider a small 2-D example to illustrate these programs. A 2-D example is simpler to explain and visualize in a short paper. Figure 7 shows the first three realizations of 100 of a binary lithofacies model. For illustration, consider the black pixels as *net* and the white as *non-net*. The realizations were generated with a sequential indicator simulation procedure (Gómez-Hernández and Srivastava, 1990; Deutsch and Journel, 1992) with an anisotropic variogram. The target proportion of *net* in the indicator simulation was 40% in all cases. Also, all realizations were constrained to a *net* datum in the exact center of the map.

Program `geo_obj` was run with all 100 realizations. Figure 8 shows the geo-objects corresponding to the first three realizations. Note the second realization where 2561 (98.5%) of the net cells are connected in one large geo-object. 1498 (67.9%) 806 (31.4%) are connected in the first and third object respectively. These numbers are read directly from the `geo_obj` summary file.

The `rank_obj` program was used to rank the realizations according to the cumulative number of cells in the first two geo-objects. Figure 9 shows the high, median, and low ranking geo-object distributions (realizations number 84, 97, and 15 respectively). For comparison, realizations 1, 2, and 3 shown earlier have rank positions of 5, 37, and 68.

The `rank_loc` was used to rank the realizations according to what is connected to the central well location. Not surprisingly, since this location is the center of the model and it was constrained to be in *net*, this ranking is nearly identical to ranking according to the cumulative number of cells in the first two geo-objects.

The `rank2loc` program can be used to rank according to the connected volume between any two well locations and also to establish the probability of two locations being connected. For example, if we rank according to the connected volume between the central well location and a location directly North, i.e., a point at the top center of the map we find that there are 11 out of 100 realizations where these two points are connected. There are 19 out of 100 realizations where the central location is connected to a point on the Eastern boundary (far right edge). Realization 84 (see Figure 9) is the highest ranking realization.

Discussion

3-D visualization tools are perhaps the most useful approach for understanding, screening, and ranking 3-D numerical geological models. There are times, however, when quantitative measures of connectivity and tortuosity are useful: the programs presented here serve this purpose.

The programs are available by anonymous FTP from the server at IAMG.ORG.

References

Deutsch, C. and Journel, A, 1992, GSLIB: Geostatistical software Library and User's Guide, Oxford University Press, New York, 340 p.

Gómez-Hernández, J.J. and Srivastava, R.M., 1990, ISIM3D: An ANSI-C Three Dimensional Multiple Indicator Conditional Simulation Program, Computers & Geosciences, Vol 16. No. 4, pp. 395-410.

Gutteridge, P. A. and Gawith, D. E., 1996, "Connected Volume Calibration for Well Path Ranking," SPE Paper Number 35503 presented at the European 3-D Reservoir Modelling Conference, Stavanger, Norway, pp. 197-206.

Mehlhorn, K., 1984, Multi-Dimensional Searching and Computational Geometry, Springer-Verlag, New York

Preparata, F. P. and Shamos, M.E., 1988, Computational Geometry: An introduction, Springer-Verlag, New York

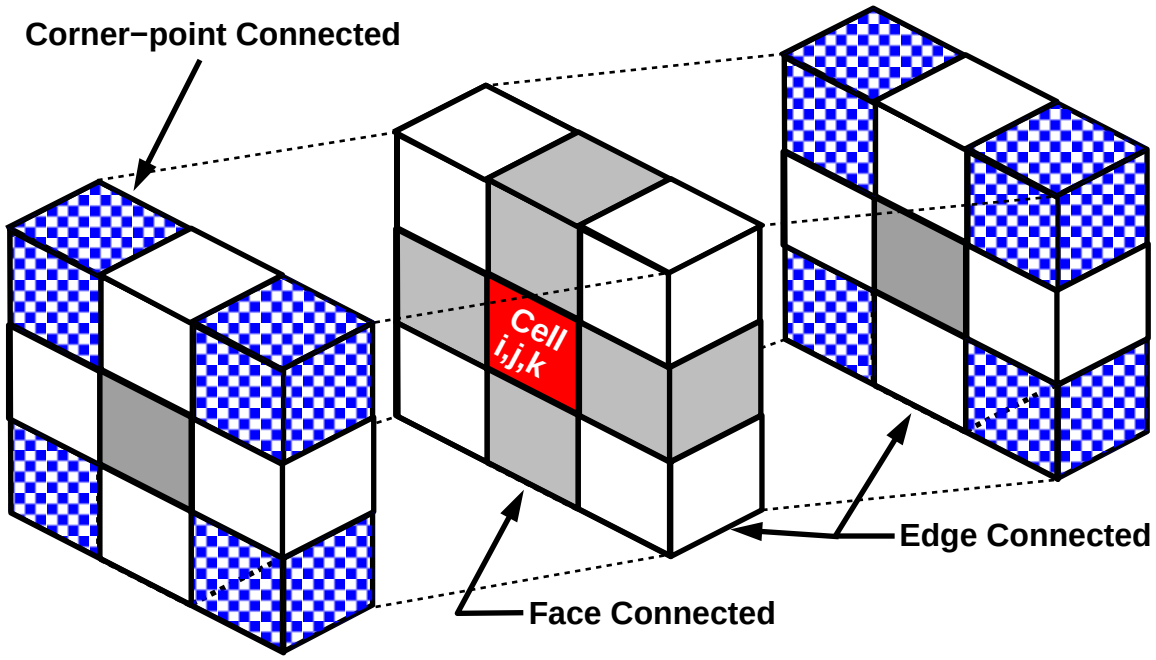

Figure 1: Illustration of the 26 cells connected to cell i, j, k. There are 6 *face*-connected cells, 12 *edge*-connected cells, and 8 *corner-point*-connected cells.

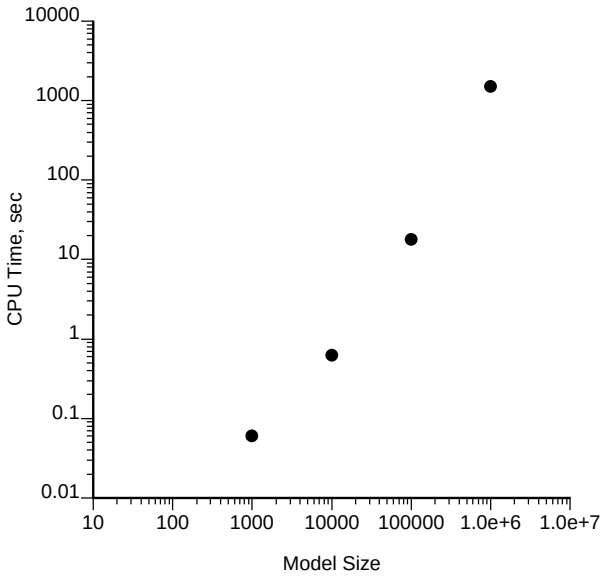

Figure 2: Plot of required CPU time (SGI Indigo2) versus model size for models between 1 thousand cells and 1 million cells.

```
                    Parameters for GEO_OBJ
                    **********************

START OF PARAMETERS:
nofile                          -Input  rock type model
sisim.out                       -Input  porosity  model
nofile                          -Input  perm      model
geo-object.model                -Output GEO-OBJECT model
geo-object.stats                -Output Stats file
50   100    1   1               -nx, ny, nz, num realizations
0  0                            -Edge / Corner Connections? (0=no,1=yes)
1  0                            -Net Rock Types: num, codes
 0.5                            -Porosity Cutoff
 0.1                            -Permeability Cutoff
```

Figure 3: Example Parameter file for `geo_obj`.

```
                  Parameters for RANK_OBJ
                  ***********************

START OF PARAMETERS:
geo-object.stats          -Input GET-OBJ stats file
rank_obj.out              -Output Ranking file
5                         -num of OBJ: cum PV ranking
5                         -num of OBJ: cum net fraction ranking
1                         -num of OBJ: tortuosity ranking
```

Figure 4: Example Parameter file for `rank_obj`.

```
                Parameters for RANK_LOC
                ***********************

START OF PARAMETERS:
geo_obj.mod               -Input GEO_OBJ model file
rank_loc.out              -Output Ranking file
100                       -number of realizations
200   1  50               -nx,   ny,   nz
1.0  1.0 1.0              -xsiz, ysiz, zsiz
50.0                      -maximum radius for connection
1                         -number of wells
5                         -  number of blocks for well i
100   1    1              -     x,y,z location
100   1    2              -     x,y,z location
100   1    3              -     x,y,z location
100   1    4              -     x,y,z location
100   1    5              -     x,y,z location
```

Figure 5: Example Parameter file for `rank_loc`.

```
                Parameters for RANK2LOC
                ***********************

START OF PARAMETERS:
geo_obj.out               -Input GEO_OBJ model file
rank2loc.out              -Output Ranking file
100                       -number of realizations
200   1  50               -nx,   ny,   nz
1.0  1.0 1.0              -xsiz, ysiz, zsiz
50.0                      -maximum departure distance
1                         -number of well pairs
5                         -  number of blocks for first well i
75    1    1              -     x,y,z location
75    1    2              -     x,y,z location
75    1    3              -     x,y,z location
75    1    4              -     x,y,z location
75    1    5              -     x,y,z location
5                         -  number of blocks for first well i
125   1    1              -     x,y,z location
125   1    2              -     x,y,z location
125   1    3              -     x,y,z location
125   1    4              -     x,y,z location
125   1    5              -     x,y,z location
```

Figure 6: Example Parameter file for `rank2loc`.

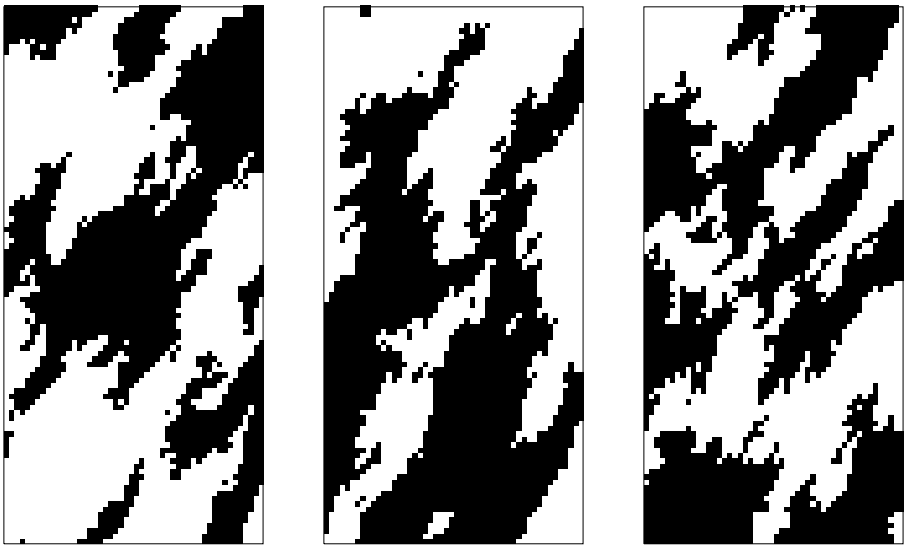

Figure 7: Three realizations of an indicator simulation process.

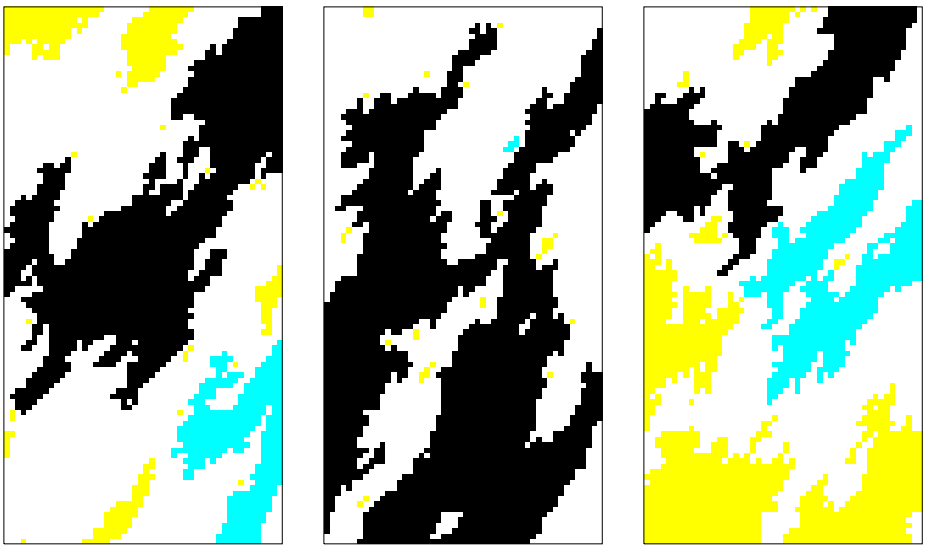

Figure 8: The three geo-object distributions corresponding to the three realizations shown on Figure 7.

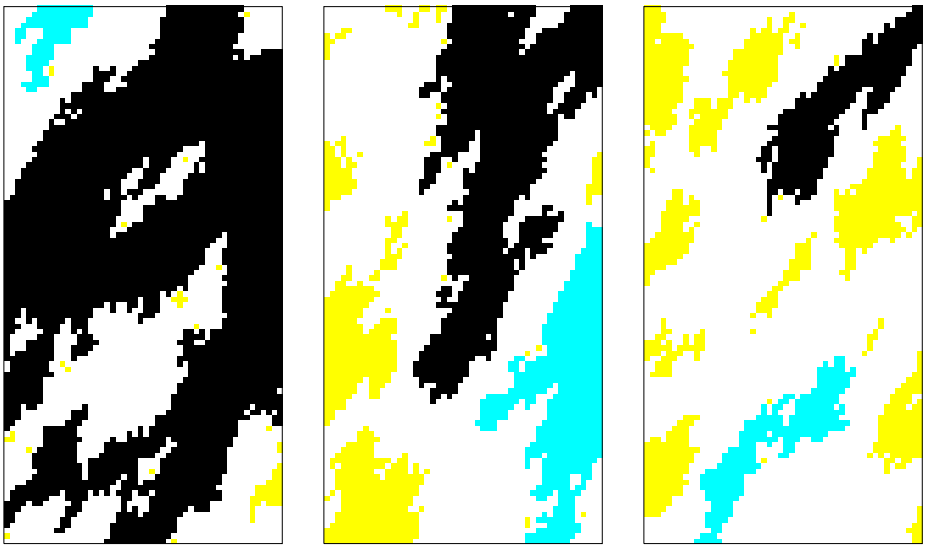

Figure 9: The high, median, and low ranking geo-object distributions (realizations number 84, 97, and 15 respectively).