Geostat utilities guide

v. 1.0

November 2013

Gregoire Mariethoz

Gregoire.mariethoz@minds.ch

1. Input and output text files

1.1. Files names

All input and output files are is a ASCII SGeMS compatible format format. Input files are saved in a file whose name prefix is defined in the params.dat file. **Files names should never contain spaces.**

1.2. Grids

Input and output grids have a GSlib format, compatible with SGeMS.

It is as follows:

- The first line consists of the size (X Y Z) of the grid, each number separated by a space.
- The second line is the number of variables present on the grid.
- Then, each line states the name of one variable: third line of the file for variable 0, fourth line for variable 1, etc.
- The rest of the file consists in columns, with one column per variable. Numbering of the grid nodes starts from the point of coordinates (1,1,1) and numbering goes by increasing coordinates starting from X, then Y and Z.
- Unknown value code is -999999999 (if some of the variables are unknown at a certain point).
- Mask value code is -1e10 (value not considered for simulation or as training data).

Example:

```
10 5 2                        //the grid is 10 by 5 by 2 nodes.
2                             //it contains 2 variables.
lithofacies                   //the name of the first variable is "lithofacies".
tomography                    //the name of the second variable is "tomography".
1 0.34                        //the node [1 1 1] has a value 1 for variable 1 and
                              //0.34 for variable 2.
2 7.8                         //the node [2 1 1] has a value 2 for variable 1 and 7.8
                              //for variable 2.
1 2.31                        //and so on.
...
etc
```

1.3. Pointsets

The format of the conditioning data consists in a set of points. It is consistent with the GSlib and SGeMS formats and is as follows:

- A first line specifying the number of conditioning data points.
- Then the number of variables, including X,Y and Z coordinates. This is the number of variables used in the simulation+3. For example, if 2 variables are simulated, this number should be 5 (x, y, z, variable1, variable2).
- Then the 3 lines:
- x
- y

- z
- Then the name of each variable, one per line.
- The rest of the file is organized in 3 columns for coordinates, plus one column per variable.
- Each line represents a point, with X Y Z V1 V2 ... Vn.
- Unknown value code is -999999999 (if some of the variables are unknown at a certain point).
- Mask value code is -1e10 (value not considered for simulation or as training data).

Example:

```
51                             //there are 51 points.
5                              //it contains 5 variables (x, y, z and 2 variables).
x                              //name of first variable (x coordinate).
y                              //name of second variable (y coordinate).
z                              //name of third variable (z coordinate).
facies                         //name of fourth variable.
porosity                       //name of fifth variable.
21.4 7.2 1 1 0.12              //the first point is at coordinates [21.4 7.2 1] and
                               //has a value 1 for variable 1 and 0.12 for variable 2.
12.1 -5.43 1 0 -999999999      //here, facies is informed, but not porosity.
...
Etc
```

2. Matlab® script utilities

All utilities are given as functions, and therefore can be used from the command line or called from scripts. The folder containing the utilities has to be added to the path of Matlab (`file>set path`…).

A grid (denoted `grid` in the utilities) is a matrix representing a Cartesian grid. Grid matrixes have 4 dimensions: `X`, `Y`, `Z` and `V`, where `V` represents the different variables located at each location of the grid. For example, an bivariate 2D grid of 100 by 150 nodes will be represented by a matrix of size [150 100 1 2] (note that Matlab reverses `X` and `Y` coordinates, but the utilities take account of this and display the grids correctly).

By itself, a grid is not spatially located and the grid elements are square. Coordinates matrixes (denoted `xx`, `yy` and `zz` in the utilities) can be added to provide the actual location of each grid cell in a Cartesian space. Such matrixes can be created with the Matlab function `meshgrid` or with the provided utility `LocateGrid`.

A pointset (denoted `pts` in the utilities) is a matrix of `npts` rows (one row per point, where `npts` denotes the number of points of the pointset) and `nvar+3` colums: `X Y Z` V_1 … V_n (`nvar` denotes the number of variables).

2.1.I/O functions and files operations

LoadGrid.m

```
Syntax:
       [grid,x,y,z,nbvar,namevar] = LoadGrid(filename)
Description:
       Function that loads a regular grid in the SGEMS (DS) format.
Output arguments:
       grid: loaded grid matrix.
       x, y, z: size of the loaded grid.
       nbvar: number of variables of the loaded grid.
       namevar: cell array containing the name of all loaded variables.
Input arguments:
       filename: the file name of the grid file to load.
```

LoadPts.m

```
Syntax:
       [pts,nbvar,namevar] = LoadPts(filename)
Description:
       Function to load a pointset in the SGEMS (DS) format.
Output arguments:
       pts: loaded matrix containing the loaded pointset.
       nbvar: number of variables of the loaded grid.
       namevar: cell array containing the name of all loaded variables.
Input arguments:
       filename: the file name of the points file to load.
```

WriteGrid.m

```
Syntax:
      WriteGrid(grid,filename,namevar)
Description:
      Function to write a grid in a SGEMS (DS) format.
Input arguments:
      grid: the grid to write.
      filename: the name of the file written.
      namevar: cell array containing the variables names.
% Function: WriteParamsFile
% Usage: WriteParamsFile(params,filename,2)
% ----------------------------------------
% Input: param - A struct of the form resulting from calling ReadParamsFile
%         filename - name of new parameter file
%         N_PAR - number of parallelizations

% Matz Haugen, Stanford University, 2010
```

WritePts.m

```
Syntax:
      WritePts(pts,filename,namevar)
Description:
      Function to write a pointset in a SGEMS (DS) format.
Input arguments:
      pts: the matrix with containing the pointset.
      filename: the name of the file written.
      namevar: cell array containing the variables names.
```

2.2. Viewing functions

ViewGrid.m

```
Syntax:
      ViewGrid(grid)
Description:
      Function to view a 2D or 3D grid, ignoring its actual location in Cartesian
      space.
Input arguments:
      grid: the 2D or 3D matrix containing the grid to visualize.
```

ViewLocatedGrid.m

```
Syntax:
```

ViewLocatedGrid(grid,xo,yo,zo,dx,dy,dz)

Description:

Function to view a 2D or 3D grid, with specified origin and cell size.

Input arguments:

grid: the 2D or 3D matrix containing the grid to visualize.

xo, yo, zo: coordinates of the grid origin (lower left corner),

dx, dy, dz: Cell size.

ViewPts.m

Syntax:

ViewPts(pts,var,dotsize,marker)

Description:

Function to view a 3D pointset.

Input arguments:

pts: the matrix with n+3 columns containing the pointset to view.

var: the variable of the pointset to represent.

dotsize: size of the represented points, in pixels.

marker: the symbol to represent each point. Type help plot for a list of markers.

2.3. Data manipulation functions

ExtractSlice.m

Syntax:

pts = ExtractSlice(sim,dim,coord)

Description:

Function to extract a 2D slice from a 3D bloc. The slice has the shape of a pointset.

Output arguments:

pts: the matrix representing the pointset of the slice.

Input arguments:

sim: the matrix containing the 3D grid.

dim: the dimension where the slice is taken (1=fixed x, 2=fixed y, 3=fixed z).

coord: the coordinate of the slice.

Grid2Pts.m

Syntax:

pts = Grid2Pts(grid,xx,yy,zz)

Description:

Function to create a pointset from a grid. Useful for creating exhaustively known secondary variables.

Output arguments:

pts: the matrix containing the pointset.

Input arguments:

grid: grid from which the points have to be extracted.

xx, yy, zz: coordinates matrixes of the grid.

LocateGrid.m

Syntax:

[xx,yy,zz] = LocateGrid(grid,dx,dy,dz,ox,oy,oz)

Description:

Function to assign a coordinate system to a grid. The coordinate system consists of coordinates matrixes indicating the location of each grid node.

Output arguments:

xx, yy, zz: the coordinates matrixes.

Input arguments:

grid: the grid to locate spatially.

dx, dy, dz: the size of the grid nodes.

ox, oy, oz: the coordinates of the origin of the grid.

NewGrid.m

Syntax:

grid = NewGrid(x,y,z,nbvar,initialvalue)

Description:

Function to create a new, non-located grid. All cells of the grid are initialized to the value initialvalue. The grid can be located spatially using the LocateGrid function.

Output arguments:

grid: the grid to create.

Input arguments:

x, y, z: size of the new grid.

nbvar: number of variables of the new grid.

initialvalue: the initial value attributed to each grid cell.

Pts2Grid.m

Syntax:

grid=Pts2Grid(pts,grid,xx,yy,zz)

Description:

Function to migrate a pointset on a grid.

Output arguments:

grid: the grid with the migrtard points included.

Input arguments:

pts: the matrix containing the pointset.

grid: the grid where the pointset has to be migrated.

xx,yy,zz: coordinates matrixes corresponding to the grid.

SampleGrid.m

Syntax:

pts=SampleGrid(grid,samplesize)

Description:

Function to randomly sample points from a grid.

Output arguments:

pts: the matrix containing the sampled pointset.

Input arguments:

grid: grid from which the points have to be sampled.

samplesize: the number of points to sample.

SelectPts.m

Syntax:

pts=SelectPts(grid,xx,yy)

Description:

Function to select manually points on a univariate 2D grid and output them as a pointset.

Output arguments:

pts: the matrix containing the selected pointset.

Input arguments:

grid: the the grid from which the pointset has to be extracted.

xxgriy,yygrid: matrixes of coordinates corresponding to the grid, as created by meshgrid.

YXZ2XYZ.m

Syntax:

grid = YXZ2XYZ(grid)

Description:

Function to change the nodes numbering of a grid from +Y +X +Z to +X +Y +Z. In Matlab the coordinates of a matrix start with increasing y, then increasing x. This function allows transforming grids so that the mapping of the matrix is coherent with the coordinated mapping in the Cartesian space.

Output arguments:

grid: the reversed grid.

Input arguments:

grid: the grid to reverse.

2.4. Simulation / estimation functions

AdjustVario.m

Syntax:

par=AdjustVario(h,g,par0,type)

Description:

Automatically adjusts variogram model on an experimental variogram.

Output arguments:

par: The adjusted variogram parameters.

Input arguments:

h: vector containing the lag distances at which the experimental variogram is known.

g: vector containing the gamma(h) values for the experimental variogram.

par0: Initial guess for the variogram parameters.

type: Type of the variogram to adjust.

ConnectFct.m

Syntax:

connectfct=ConnectFct(grid,var,show)

Description:

Computes the connectivity function of an image given as a grid. Note that this function uses the file /utils/MATLAB_BIN_UTILS/connectfct_parallel.exe. If the file connectfct_parallel.exe is located in a different directory, the function ConnectFct.m should be corrected to indicate the correct path.

Output arguments:

connectfct: structure containing 3 fields: x, y and z, each containing two columns constituting the function h, P(connection,h).

Input arguments:

grid: Input grid.

var: Variable on which to compute the connectivity function.

show: 1 = display connectivity functions, 0 = do not display.

ConnectMap.m

Syntax:

connectmap=ConnectMap(grid,var,facies)

Description:

Computes the geobody map of an image for a given facies. Note that this function uses the file /utils/MATLAB_BIN_UTILS/connectmap.exe. If the file connectmap.exe is located in a different directory, the function ConnectMap.m should be corrected to indicate the correct path.

Output arguments:

connectmap: grid of the same size as grid, with the geobody code at each node.

Input arguments:

grid: Input grid.

var: Variable on which to compute the connectivity function.

facies: Facies on which connectivity will be computed.

DrawVariogram.m

Syntax:

DrawVariogram(meanh,range,sill,param3,type,nugget)

Description:

Draws a variogram model at specified lag distances. Meant to be used for variogram adjustments.

Input arguments:

meanh: lag distances at which gamma(h) should be calculated. These are usually the output of the function Variogram.m.

range: Variogram range.

sill: Variogram sill.

param3: Variogram third parameters.

type: variogram type. 1=exponential, 2=Gaussian. 3=spherical, 4=power, 5=cardinal sine, 6=linear.nugget: Variogram nugget effect.

nugget: nugget effect.

KrigptOrd.m

Syntax:

[v0,s0]=KrigptOrd(x,y,v,x0,y0,a,sill,param3,m,nugget,nn)

Description:

Estimates the value of a single point using ordinary kriging (OK), following the book of Isaac and Srivastava (pp. 291-296). Written by Philippe Renard, modified by G. Mariethoz

Output arguments:

v0: Estimated kriging mean.

s0: Estimated kriging variance.

Input arguments:

x, y: vectors defining the coordinates of the neighbor informed locations.

v: vector of values at the neighbor informed locations.

x0, y0: Coordinates of the location to estimate.

a: Variogram range.

sill: Variogram sill.

param3: Variogram third parameters.

m: covariance model. 1=exponential, 2=Gaussian. 3=spherical, 4=power, 5=cardinal sine, 6=linear.

nugget: Variogram nugget effect.

nn: Number of closest neighbors to consider for the estimation. If more neighbors are present, they will be ignored.

KrigptSimple.m

Syntax:

[v0,s0]= KrigptSimple(x,y,v,mu,x0,y0,a,sill,param3,m,nugget,nn)

Description:

Estimates the value of a single point using simple kriging (SK).

Output arguments:

v0: Estimated kriging mean.

s0: Estimated kriging variance.

Input arguments:

x, y: vectors defining the coordinates of the neighbor informed locations.

v: vector of values at the neighbor informed locations.

mu: Mean of the random variable to simulate.

x0, y0: Coordinates of the location to estimate.

a: Variogram range.

sill: Variogram sill.

param3: Variogram third parameters.

m: covariance model. 1=exponential, 2=Gaussian. 3=spherical, 4=power, 5=cardinal sine, 6=linear.nugget: Variogram nugget effect.

nn: Number of closest neighbors to consider for the estimation. If more neighbors are present, they will be ignored.

Ksmooth.m

Syntax:

[pdf,xx,yy]=Ksmooth(Y, sig, Xspacing, Yspacing)

Description:

Performs density estimation by kernel smoothing in 2D (or 1D) using Gaussian kernels.

Output arguments:

pdf: the grid of density estimates.

xx, yy: Coordinates matrixes of the density estimation grid.

Input arguments:

Y: 2-column matrix containing the x and y coordinates of the points of occurrence

sig: Standard deviation of the Gaussian kernels.

Xspacing, Yspacing: size of the elements on the density grid.

MGSimulFFT.m

Syntax:

grid=MGSimulFFT(x,y,z,mu,sigma2,m,lx,ly,lz)

Description:

Generates one unconditional realization of a multiGaussian random function using FFT. It is based on a code originally written by Olaf Cirpka.

Output arguments:

grid: the grid containing the generated multiGaussian variable.

Input arguments:

x, y, z: the size of the simulation grid.

mu: mean of the simulation.

sigma2: variance of the simulation.

m: covariance model. 1=exponential covariance. 2=Gaussian covariance.

lx, ly, lz: correlation lengths. The units are in number of grid nodes.

MGSimulFFTCond.m

Syntax:

`grid=MGSimulFFTCond(x,y,pts,mu,sigma2,m,lx,nn)`

Description:

Generates one conditional realization of a multiGaussian random function using FFT, and conditioning to data using simple kriging. It is restricted to 2D grids.

Output arguments:

grid: the grid containing the generated multiGaussian variable.

Input arguments:

x, y: the size of the simulation grid.

pts: matrix containing the conditioning data.

mu: mean of the simulation.

sigma2: variance of the simulation.

m: covariance model. 0=exponential covariance. 1=Gaussian covariance.

lx: correlation lengths (isotropic). The units are in number of grid nodes.

nn: Number of closest neighbors to consider for kriging. If more neighbors are present, they will be ignored.

MGSimulSGS.m

Syntax:

`grid=MGSimulSGS(x,y,pts,mu,sigma2,m,lx,nn)`

Description:

Generates one conditional realization of a multiGaussian random function using SGS. It is restricted to 2D grids.

Output arguments:

grid: the grid containing the generated multiGaussian variable.

Input arguments:

x, y: the size of the simulation grid.

pts: matrix containing the conditioning data.

mu: mean of the simulation.

sigma2: variance of the simulation.

m: covariance model. 1=exponential, 2=Gaussian. 3=spherical, 4=power, 5=cardinal sine, 6=linear.nugget: Variogram nugget effect.

lx: correlation lengths (isotropic). The units are in number of grid nodes.

nn: Number of closest neighbors to consider for simulating each node. If more neighbors are present, they will be ignored.

MovAvg.m

Syntax:

`movavg=MovAvg(grid,window)`

Description:

Function to compute a window moving average on a 1D, 2D or 3D grid.

Output arguments:

movavg: the input grid.

Input arguments:

grid: the resulting smoothed grid.

window: the window size defined as the number of pixels in each direction from a central pixel (therefore this number in fact corresponds to half the window).

Variogram.m

Syntax:

[meanh,gammah]=Variogram(pts,nblags,maxh,sampling,draw)

Description:

Function to compute the experimental variogram of a pointset.

Output arguments:

meanh: the center of each lag distance class.

gammah: the variogram value for each lag distance class.

Input arguments:

pts: the matrix containing the pointset. If it contains more than one variables, the variogram is computed from the first variable.

nblags: number of lags for the variogram computation.

maxh: maximum distance for the variogram computation, expressed in spatial units.

sampling: to improve the speed of the variogram calculation, it is possible to use only a subset of the pointset. This values determine how this sampling will be done. For example, a value 5 considers 1/5 of the total pointset.

draw: 1=represent the variogram graphically. 0=do not represent it.

CVariogram.m

Syntax:

[meanh,gammah,cvar]=Variogram(pts,nblags,maxh,sampling,draw)

Description:

Function to compute the experimental cross-variogram of a pointset.

Output arguments:

meanh: the center of each lag distance class.

gammah: the cross-variogram value for each lag distance class.

cvar: the cross terms of the covariance matrix

Input arguments:

pts: the matrix containing the pointset. It has to contain 2 variables (therefore 5 columns).

nblags: number of lags for the variogram computation.

maxh: maximum distance for the variogram computation, expressed in spatial units.

sampling: to improve the speed of the variogram calculation, it is possible to use only a subset of the pointset. This values determine how this sampling will be done. For example, a value 5 considers 1/5 of the total pointset.

draw: 1=represent the variogram graphically. 0=do not represent it.