HYDRO_GEN

A New Method for the Generation of Random Functions

Code Description and User's Guide

ALBERTO BELLIN[1] and YORAM RUBIN[2]

1. Dipartimento di Ingegneria Civile ed Ambientale
Università di Trento, I-38050 Povo-Trento, Italy
phone: +39 461 882620; fax: +39 461 882672
e-mail: Alberto.Bellin@ing.unitn.it

2. Department of Civil Engineering
University of California, Berkeley, CA 94720 USA
phone: +1 510 642 2282; fax: +1 510 642 7476
e-mail: rubin@ce.berkeley.edu

HYDRO_GEN license information

Please cite the following paper in publications concerning unconditional generation of stationary random functions: Bellin A., Y. Rubin, HYDRO_GEN: A new random field generator for correlated properties, *Stochastic Hydrology and Hydraulics, 10(4), 1996* and the following paper for conditional and unconditional generation of fractal random fields: Rubin, Y. and A. Bellin, Conditional Simulation of Geologic media with Evolving Scales of Heterogeneity, In: *Scale dependence and Scale Invariance in Hydrology*, ed. G. Sposito, Cambridge University Press, 1997, (in press).

disclaimer

Flannery, S. A. Teukolsky, W. T. Vetterling, for the function computing the Bessel Function

Copyright conditions of the above referenced libraries are extended to HYDRO_GEN.

1 Introduction

Hydro_gen (Bellin and Rubin, 1996; Rubin and Bellin, 1997) is a computer code for generating two-dimensional space random functions with an assigned covariance structure. The code is written in Ansi Fortran 77 with a quite standard implementation that allows the use of a wide class of computers. The computations are performed in double precision and the actual configuration has been tested on the workstation IBM RISC/6000 mod. 320H.

To create the executable file, simply compile and link together the source files distributed with this manual, keeping in mind that the main program is in the file called hydro_gen.f.

An example of a compiling session on a Unix machine is:

```
f77 -c -O *.f
f77 -o hydro_gen *.o
```

The flags can assume different meanings on different Unix systems. For the Fortran compiler (IBM AIX XL FORTRAN Compiler/6000) running on the workstation IBM RISC/6000 mod. 320H, they are:

-c instruct the compiler that the file must be only compiled and not linked;

-O instruct the compiler to perform an optimized compilation. The user can usually choose between different kinds of optimizers; we suggest the optimizer aimed to increase the speed of program execution;

-o instruct the compiler about the name of the executable file.

2 Code organization

The code contains the following files:

hydro_gen.f : the main program containing the definitions of the arrays and variables, the calls for the computation of the kriging coefficients, the selection of the path followed during the field generation, the calls for the linear combination of suitable kriging points, and the generation of the field;

coefcy.f : subroutine which computes the kriging coefficients used in the coarse grid generation step and in the first level of the refinement process;

coefl2.f : subroutine which computes the kriging coefficients used in the second level of the refinement process;

comb.f : subroutine which computes the conditional mean at the generic point $\mathbf{x}$;

covariance.f : subroutine which computes the covariance matrix with reference to the larger search neighborhood;

covar.f : collection of covariance function options. This file can be modified by users who would like to use covariance functions different from the preloaded ones. The calling command for the function is $cov(itype, r_x, r_y)$ where $itype$ identifies the covariance function and r_x,r_y are the lag distances at which the covariance is computed. The following options are possible:

$itype = 0$ discrete covariance function. It is read from a file;

itype=1 exponential covariance function: $C_Z(r_x, r_y) = e^{-r'}$, where $r' = \sqrt{(r_x/I_{Z_x})^2 + (r_y/I_{Z_y})^2}$ and $r_x = x_2 - x_1$ and $r_y = y_2 - y_1$ are the x and y components of the two–point separation lag. I_{Z_x} and I_{Z_y} are the integral scales in the x and y directions, respectively;

$itype = 2$ Gaussian covariance function: $C_Z(r) = e^{-r'^2}$;

$itype = 3$ Whittle isotropic covariance function: $C_Z(r_x, r_y) = \alpha r K_1(\alpha r)$, where $r = \sqrt{r_x^2 + r_y^2}$, $\alpha = \pi/2I_Z$ with $I_{Z_x} = I_{Z_y} = I_Z$ and K_1 is the first–order modified Bessel function of the third kind;

$itype = 4$ Mizell isotropic covariance function (type B) [Mizell, 1982]: $C_Z(r_x, r_y) = \alpha r K_1(\alpha r) - (\alpha r)^2 K_0(\alpha r)$, with the same symbols of the previous covariance function and K_0 the modified zero–order Bessel function of the third kind;

$itype = 5$ Power law semivariogram: $\gamma_Z(r_x, r_y) = c\ r^\beta$, with $r_x = x_2 - x_1$ and $r_y = y_2 - y_1$ are the x and y components of the two point separation lag and $r = \sqrt{(r_x/I_x)^2 + (r_y^2/I_y)^2}$. The exponent β isrelated to the Hurst exponent H by the following relationship: $\beta = 2\ H$. Furthermore I_x and I_y are the reference scales along the directions x and y, respectively.

bessik.f : functions that compute the modified zero– and first–order Bessel functions of third kind.

Additionally, the following files contain mathematical and statistical library routines as described (see also the main program introductory comments):

linpack.f : contains the Linpack library subroutines dspfa (to factor a double precision packed symmetric matrix) and dspsl (to solve the double precision symmetric system a*x=b using the factors computed by dspfa);

blas.f : contains the Blas library subroutines and functions daxpy, dswap, idamax, and ddot. These are linear algebra routines used by the Linpack routines;

ranlib.f : contains the necessary subroutines and functions from the Ranlib package to initialize the seed for and to generate normally distributed random variables.

The above routines can be replaced by corresponding routines from different packages . In that case, however, the authors do not guarantee the accuracy of the reproduced field. Specifically, the generation of the normally distributed random values requires a particular attention. The user intending to substitute the Ranlib package is strongly advised to perform tests for normality of the generated numbers and convergence of the mean and variance.

According to the user's choice the generation is performed in two ways: 1) direct generation on the selected grid; 2) generation over a coarse grid followed by up to three nested refinements. At each refinement step the block size is reduced by a factor of 2. We advise the user to respect the limit of three nested refinement levels in order to avoid appreciable reduction in accuracy. During the input session the user will be required to provide the number of nested refinement levels. Note that 0 reference levels implies no refinement.

The refinement step can be useful when the search neighborhood area is large. In these situations the computational effort for the coarse grid generation can be reduced without an appreciable impact on the accuracy.

At the beginning of the main program, values of integer variables are assigned to reserve space for the arrays used in the code. These variables are:

$igrid1$: maximum number of grid points in the x direction.

$igrid2$: maximum number of grid points in the y direction. These variables are used to reserve space for the array $cond(i,j), i = 1,...,igrid1; j = 1,...,igrid2$, which contains the generated random field. The maximum field dimensions that can be handled, then, are $igrid1 * \Delta x$ by $igrid2 * \Delta y$, where Δx and Δy are the discretizations of the grid. For fractal fields the larger search neighborhood coincides with the generation domain.

$icond$: maximum number of points inside the search neighborhood area that can be used to compute the conditional mean and variance. It is used to reserve space for the kriging array $ap(icond * (icond + 1)/2)$, which is stored in packed format.

$iptmx$: maximum dimension of the array storing the independent kriging coefficients. The exact number of independent kriging coefficients is computed at each run and printed to standard output.

$iptmxcv$: maximum dimension of the vector storing the independent conditional variances. In analogy with the previous point the exact number of vector positions required at each run is computed and printed to standard output.

$idim$: maximum dimension of the matrix containing the covariance function. In each run the dimensions are $(NN2 + NN2A) * NN1$. For example, $NN2 = xsp/\Delta x$, where xsp is the size of the search neighborhood dimension as shown in Figure 1 and Δx is the coarse grid x–direction discretization. In case of fractal fields $NN2$ and $NN1$ are the number of coarse grid nodes in x and y directions while $NN2A = 0$.

The main arrays used are:

$cond(i,j)$: the field values in each realization;

$u1sid(i)$: the independent kriging coefficients;

$vcxsid(i)$: the independent conditional variances;

$r(i)$: the normally distributed variables with mean $\langle Z \rangle = 0$ and variance $\sigma_Z^2 = 1$;

$nseed$: the seed for the generation of the vector r. At the beginning of each run, the user is prompted for either manual or automatic (via the computer clock) seed assignment (see the following section).

3 Description of the input variables

The main advantage of the program's method is that the kriging coefficients depend on the field geometry and grid spacing but not on the actual field values. To capitalize on this advantage the user can choose between two options:

1 compute the interpolation coefficients and store them in a file

2 read the interpolation coefficients from the file where previously generated interpolation coefficients have been stored

The option 2 is selected when the interpolation coefficients have been computed in a previus simulation usung the same grid spacing and autocorrelation function. For additional information we refer to the example Section of this document.

$gset$: decision variable for setting the random number generator seed. If y, the user is prompted for the seed number; if any other letter, the clock is used to set the seed. The code for the latter option is written for use with Unix systems and must be modified by DOS users (see the main program at the line labeled 1001);

dx, dy : grid dimension in the x and y directions;

np : number of independent monte carlo replicates;

l_x, l_y : field dimensions;

$sigy$: field variance (the unconditional one);

$cond10$: mean (the unconditional one);

$imark$: integer variable which indicates whether the kriging coefficients have already been computed and stored in the file $filecoef$ (from a previous run; option 2 above) or need to be computed and stored (option 1 above). A value of 1 indicates that the values have already been computed and stored, and any other value indicates otherwise. (Note: the coefficients are stored in the file in unformatted binary format.);

$itype$: integer variable which indicates the type of autocorrelation function $\rho(r_x, r_y) = C(r_x, r_y)/\sigma_Z^2$ employed (see Section 2 for the options). r_x and r_y are the x and y components of the distance between the two points between which the covariance function C_Z is computed, and σ_Z^2 is the variance. If itype=0 the autocovariance is read from a file specified by the user. The file must contain the discretized autocovariance function on a subgrid of dimensions $(xsp + xspa) \times ysp$ with grid spacing equal to the coarse grid spacing. The matrix is written in free format row by row without empty lines (see Appendix A). The discretized covariance function used for the two levels of refinement must be introduced immediately after the coarse grid covariance function in the following order (see Appendix A):

Level No. 1

$\rho(0,0) \quad \rho(\Delta x_1, 0)$

$\rho(0, \Delta x_2) \quad \rho(\Delta x_1, \Delta x_2)$

$\rho(-\Delta x_1/2, -\Delta x_2/2) \quad \rho(\Delta x_1/2, -\Delta x_2/2) \quad \rho(-\Delta x_1/2, \Delta x_2/2) \quad \rho(\Delta x_1/2, \Delta x_2/2)$

While the data for the second level of refinement are:

$\rho(0,0)$

$\rho(\Delta x_1/2, -\Delta x_2/2)$

$\rho(\Delta x_1/2, 0)$

$\rho(\Delta x_1/2, \Delta x_2/2)$

$\rho(0, \Delta x_2)$

$\rho(-\Delta x_1/2, \Delta x_2/2)$

$\rho(-\Delta x_1/2, 0)$

$\rho(0, -\Delta x_2/2)$

$\rho(\Delta x_1/2, 0)$

$\rho(0, \Delta x_2/2)$

$sclx, scly$: integral scales in the x and y directions. The integral scales are read only if the autocovariance function is given analytically. In the case of an isotropic function, sclx=scly and only one value is read. For the fractal fields sclx and scly assume the meaning of reference scales since integral scales cannot bcannot be defined.

xsp, ysp : main dimensions of the search neighborhood area in the x and y directions (see Figure 1). Suggested values, based on the chosen covariance function and integral/correlation scales, are calculated and printed to standard output. The search neighborhood dimensions must be reasonably smaller than the field dimensions or a segmentation fault will result. The code tests for this possibility and issues a warning to the user, who may then opt to enter search neighborhood dimensions smaller than those suggested by the author or to stop the program and begin again with a larger field. Because the suggested search neighborhood dimensions are based on the authors' numerical tests for covariance structure reproduction, use of dimensions much smaller than those suggested is not recommended; it is better to work with a larger field if possible. In case of fractal fields the search neighborhood is automatically fixed equal to the field dimension. The number of nested refinement stages should be sufficiently large to avoid segmentation fault, but at the same time, enough to preserve an acceptable level of accuracy of the generated fields. The number of refinement stages depends on the ratio between the resulting coarse grid spacing and the integral scales. Numerical tests performed by the authors suggest that for regular random fields the coarse grid spacing should not exceed two integral scales while for fractal fields the limit depends on the value of β and on the field dimension. For this reason the optimal coarse grid spacing should be fixed according to tests performed by the user using different number of multistage refinements.

$xspa$: secondary dimension of the search neighborhood area in the x direction (see Figure 1). As with the main dimensions, a suggested value is calculated and printed to standard output;

$filecoef$: name of the file in which kriging coefficients are stored (if imark is 1) or should be stored once computed (if imark is not 1). Storage is in unformatted binary format;

ilevref : number of nested refinement stages. The generation is first performed on a coarse grid and then subsections of the grid are refined by increasing the density of nodes. At each refinement stage, which is performed in two steps, the grid density increases by a factor of two. As an example, to obtain a final grid spacing of 0.25, the coarse grid spacing should be 0.50 and 1.0, for one and two refinement stages respectively. The latter is more computationally efficient;

$file1$: name of the output file containing the generated fields for all Monte Carlo replicates;

$iformat$: integer variable indicated whether the field values should be output in 3–column x,y,z format (iformat=1) or matrix format (iformat=0).

In addition to the output files $filecoef$ and $file1$, the file $stats.out$, which shows the mean and variance for each replicate, is generated with each run.

4 Examples

This section describes the typical input session for the following two cases:

1. A two-dimensional anisotropic random field of the attribute Z with mean $\langle Z \rangle = 1.0$ and an exponential covariance function $C_Z(r) = \sigma_Z^2 e^{-r'}$, where σ_Z^2 is the variance of the random field Z and $r' = \sqrt{(r_x/I_x)^2 + (r_y/I_y)^2}$, with I_x and I_y the integral scales in the two coordinate directions. Here r_x and r_y are the components of the two-points lag vector;

2. A two-dimensional self-similar random field with the power law semivariogram: $\gamma_Z(r) = cr^{\beta}$; $r = \sqrt{r_x^2 + r_y^2}$, where c is a constant and $\beta = 2H$ (H is the Hurst coefficient) (Hurst, 1951).

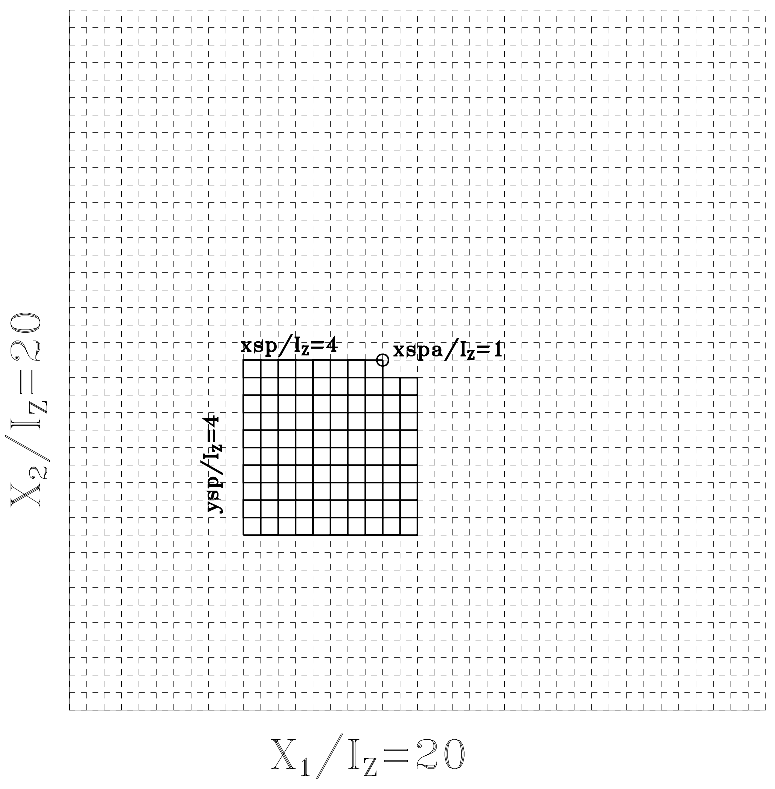

Figure 1: Example of domain discretization, grid refinement and search neighborhood area; dx=dy=0.5.

4.1 Example 1

This example shows the typical input session for the generation of realizations of a Stationary Random Space Function Z characterized by a constant mean $\langle Z \rangle$ and an exponential anisotropic covariance function:

$$C_Z(r) = \sigma_Z^2 e^{r'} \tag{1}$$

where σ_Z^2 is the variance and $r' = \sqrt{(r_x/I_x)^2 + (r_y/I_y)^2}$. Here I_x and I_y are the integral scales along the coordinate directions x and y. Distances are made dimensionless with reference to I_x.

The generated field parameters are as follows:

- field dimensions: $L_x = 40I_x$, $L_y = 4I_x$;
- integral scale along the $y-$direction: $I_y = 0.1\, I_x$. The ratio of anysotropy is then equal to $e = I_y/I_x = 0.1$;
- grid spacings: $\Delta x = 0.25I_x$; $\Delta y = 0.025I_x$;
- The imposed mean and variance are $\langle Z \rangle = 1$ and $\sigma_Z^2 = 1$, respectively.

In the following the input section for the above case is described in detail. The output appearing on the screen (requests of data and hardcopy of them) is in boldface, the data provided by the user are in italic and finally comments to the input session appear in brackets.

To initiate the execution of the program type: *hydrogen* and press return. The banner HYDRO_GEN plus the author names will appear on the screen together with the following sentence:

enter 'y' to set the seed number manually, or
enter any other letter to have the clock set the seed:
(this option is not enabled for the PC version of the code. The seed number is required in order to set up the generator of random numbers)
y
Enter the seed number as an integer between 2 and 2147483398:
14578921
SEEDS: 14578921 14578920
Enter final grid spacing dx,dy

```
.25   0.025
dx and dy .25 .03
Enter number of Monte Carlo realizations
2
# of monte carlo iterations:           2
Enter field dimensions
40.  4.
field dimensions:        40.00       4.00
To read the interpolation "kriging" coefficients
from a file enter [1]; if not,
enter any other integer:
```

(the option [1] is used to obtain new realizations of a previously generated field or to change those characteristics that do not influence the interpolation coefficient. The first case occurs when the user wants new realizations of the random field that has been the object of a previous run. The second case occurs when the user wants to generate a new random field with the same statistical properties but with different domain dimensions)

```
4
Enter the covariance type:
itype=0 ==> discrete covariance function (from file)
itype=1 ==> exponential
itype=2 ==> Gaussian
itype=3 ==> Whittle
itype=4 ==> Mizell (B)
itype=5 ==> power law semivariogram
(self similar field)
1
covariance type:Exponential
covariance of type: 1
Enter the variance
1
variance:        1.000000000000000
Enter the mean
1.
mean:        1.00
Enter integral scales in x and y directions:
1.    0.1
```

```
integral scales:        1.00        .10
search neighborhood dimensions:
--------            -
|      |            |
|      --------    ysp
|              |    |
|              |    |
---------------     -
|--xsp-|-xspa-|

Suggested values:
xsp =     3.000
ysp =     .400
xspa =     1.000
Enter xsp, ysp
3.0    0.4
xsp and ysp: 3.000 .400
Enter xspa
1.
xspa: 1.000
Enter the file name for the file storing the interpolation coefficients:
coef.dat
enter the number of refinement levels:
[0] ==> No refinement
[n] ==> refinement at n levels
1
```

(The program generates first a coarse grid with spacing $\Delta x = 0.5$, $\Delta y = 0.05$ which is then refined once in order to obtain the final grid spacing of $\Delta x = 0.25$ and $\Delta y = 0.025$)

```
Enter the name of the output file for the
replicates [max 30 characters]:
example1.dat
Enter the format of the output file:
type 1 for 3 columns x y z
type 0 for the matrix format (only z values on a regular grid)
0
```

(the program computes the interpolation (kriging) coefficients and produces the following output on the screen:

```
ngen: 26889
side #1
# of vector positions for the side #1: 279 6
side #2
# of vector positions for the side #2: 411 8
side #3
# of vector positions for the side #3: 711 16
corners:
# of vector positions for the corners: 2271 79
total # of vector positions: 2349 80
```

(Once the intepolation coefficients are computed the following sentence is issued and the program continues with the generation of the independent realizations of the RSF which are stored in the file example1.dat):

```
kriging coefficients computed; starting Monte Carlo..
```

(Once the generation of the realizations is completed the following message will appear on the screen:)

```
PROGRAM COMPLETE.
```

4.2 Example 2

This example shows the typical input session for the generation of realizations of a Self-Similar random field. Distances are made dimensionless with respect to I which represents the minimun scale of variability resolved by the simulations. The field is non-stationary and characterized by the following semivariogram:

$$\gamma_Z(r) = cr'^{\beta} \qquad (2)$$

where c is a constant related to the field variance, β is the exponent that controls how the semivariogram grows with the distance and $r' = \sqrt{(r_x/I_x)^2 + (r_y/I_y)^2}$ is the dimensionless two points lag. The exponent β is related to the Hurst coefficient through the following relationship: $\beta = 2H$. For $\beta < 1$ the field is grainy in appearance with values correlated over very short distances. This phenomenon is called antipersistence. For $\beta > 1$ correlation persists over large distances and the field is smoother in appearance but with higher contranst over large distances. This is called antipersistence.

The generated field parameters are as follows:

- field dimensions: $L_x = L_y = 100I$;
- reference scales: $I_x = I_y = I$. Note that Self-Affine Random Space Functions can be obtained using different reference scales along x and y directions;
- grid spacings: $\Delta x = I$; $\Delta y = I$. In this case no spatial variability is encountered at scales smaller than I.

To launch the program type: *hydrogen* and press return. The banner Hydro_gen will appear on the screen together with the following sentence:

```
Enter 'y' to set the seed number manually, or
enter any other letter to have the clock set the seed:
y
Enter the seed number as an integer
between 2 and 2147483398:
12345678
SEEDS: 12345678 12345677
Enter final grid spacing dx,dy
1.  1.
dx and dy 1.00 1.00
Enter number of Monte Carlo realizations
2
# of monte carlo iterations: 2
Enter field dimensions
100.  100.
field dimensions: 100.00 100.00
To read the interpolation "kriging" coefficients
from a file enter [1]; if not,
enter any other integer:
0
Enter the covariance type:
itype=0 ==> discrete covariance function (from file)
```

itype=1 ==> exponential
itype=2 ==> Gaussian
itype=3 ==> Whittle
itype=4 ==> Mizell (B)
itype=5 ==> power law semivariogram
(self similar field)
5
Covariance type:Self-Similar
covariance of type: 5
do you want to force the mean to be constant in
each realization? [Y/N]
N

(forcing the mean to a constant value in each realization is equivalent at assuming deterministic and known the large scale field variabilities)

Enter the reference scales in x and y directions:
1. 1.
reference scales: 1.00 1.00
enter the cofficients a and beta of the power
semivariogram (g=a r^beta)
Enter the file name for the file storing the
interpolation coefficients:
coefficients
enter the number of refinement levels:
[0] ==> No refinement
[n] ==> refinement at n levels
3

(the generation is performed using three nested reference levels).

Enter the name of the output file for the
replicates [max 30 characters]:
example2.dat
Enter the format of the output file:
type 1 for 3 columns x y z
type 0 for the matrix format (only z values
on a regular grid)

0 (The program computes the interpolation (kriging) coefficients. During the computation the following data are issued):

```
ngen: 10669
the semivariogram is computed with reference to
the following parameters:
C= 0.29999999999999974E-03
beta= 1.60000000000000009
FRACTAL DIMENSION: 2.200000000000000
the employed semivariogram is stored in the file:
semivariog.out
 line  nvectpos  kkcv
   1      105      14
   2      435      29
   3      990      44
   4     1770      59
   5     2775      74
   6     4005      89
   7     5460     104
   8     7140     119
   9     9045     134
  10    11175     149
  11    13530     164
  12    16110     179
  13    18915     194
  14    21945     209
  15    25200     224
```

(nvectpos is the number of vector positions required to store the interpolation coefficient and kkcv is the number of vector positions required to store the conditional variances)

```
total # of vector positions: 25200 224
```

(Once computed the intepolation coefficients the following sentence is issued and the program continues with the generation of the independent realizations of the RSF):

```
kriging coefficients computed; starting monte carlo..
```

(Once the generation of the realizations is completed the following message will appear on the screen:)

```
PROGRAM COMPLETE.
```

5 References

Bellin, A., and Y. Rubin, Hydro_gen: A new random number generator for correlated properties, *Stochastic Hydrology and Hydraulics*, 10(4), 1996.

Hurst, H. E., Long-term storage capacity of reservoirs, *Trans. Am. Soc. Civ. Eng.*, 116, 770–808, 1951.

Mizell, S.A., Gutjahr, A.L., and L.W. Gelhar, Stochastic analysis of spatial variability in two–dimensional steady groundwater flow assuming stationary and nonstationary heads, *Water Resources Research*, 18(4), 1053–1067, 1982.

Rubin, Y., and A. Bellin, Conditional Simulation of Geologic media with Evolving Scales of Heterogeneity, In: G. Sposito (ed.), *Scale Dependence and Scale Invariance in Hydrology*, Cambridge University Press, 1997, (in press).

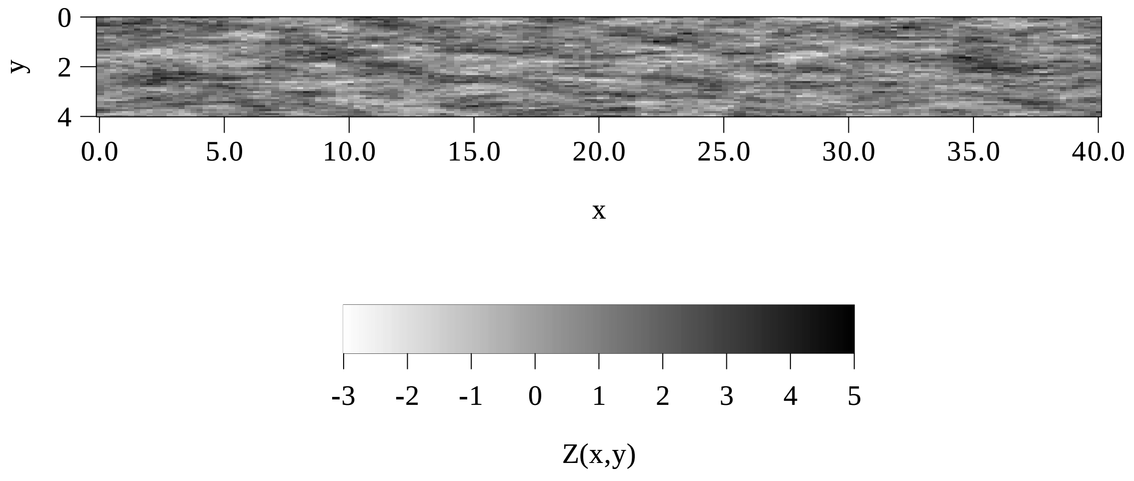

Figure 2: Example of stratified random field

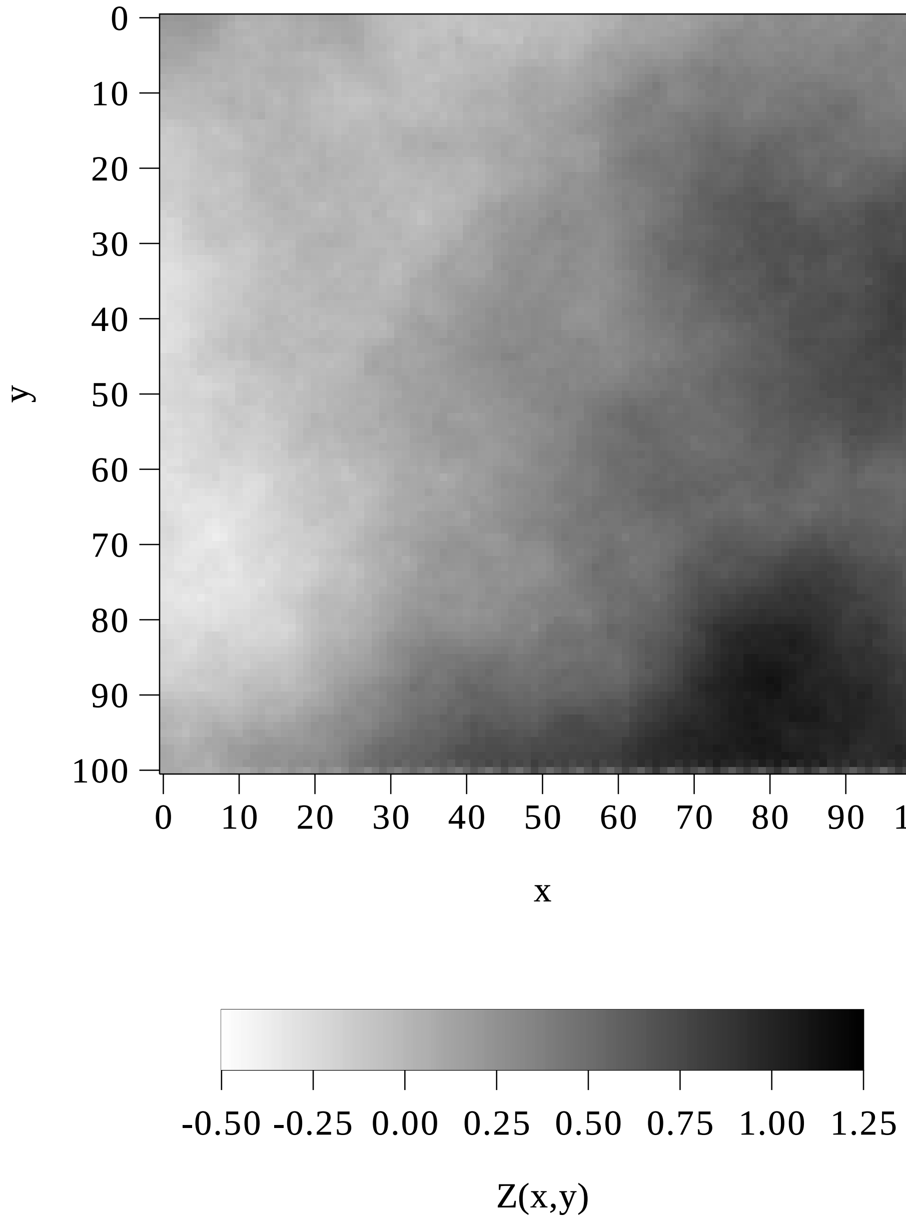

Figure 3: Example of a self Similar random field

APPENDIX A

Example of input file for a discrete covariance function. The neighborhhod area is 5x4 integral scales I, and the grid space Dx=Dy=0.5 I.

```
 1.00000  .60653  .36788  .22313  .13534  .08208  .04979  .03020  .01832  .01111  .00674
  .60653  .49307  .32692  .20574  .12726  .07812  .04777  .02914  .01775  .01081  .00657
  .36788  .32692  .24312  .16484  .10688  .06771  .04233  .02625  .01619  .00995  .00610
  .22313  .20574  .16484  .11987  .08208  .05418  .03494  .02220  .01395  .00871  .00541
  .13534  .12726  .10688  .08208  .05911  .04070  .02717  .01775  .01142  .00727  .00458
  .08208  .07812  .06771  .05418  .04070  .02914  .02014  .01355  .00894  .00581  .00373
  .04979  .04777  .04233  .03494  .02717  .02014  .01437  .00995  .00674  .00448  .00294
  .03020  .02914  .02625  .02220  .01775  .01355  .00995  .00709  .00492  .00334  .00224
  .01832  .01775  .01619  .01395  .01142  .00894  .00674  .00492  .00349  .00243  .00166
 1.00000 0.60653
 0.60653 0.49306
 0.70218 0.70218 0.70218 0.70218
 1.00000
 0.70218
 0.60653
 0.70218
 0.60653
 0.70218
 0.77880
 0.77880
 0.77880
 0.77880
```