NAG Toolbox

nag_rand_field_1d_generate (g05zp)

1 Purpose

nag_rand_field_1d_generate (g05zp) produces realizations of a stationary Gaussian random field in one dimension, using the circulant embedding method. The square roots of the eigenvalues of the extended covariance matrix (or embedding matrix) need to be input, and can be calculated using nag_rand_field_1d_user_setup (g05zm) or nag_rand_field_1d_predef_setup (g05zn).

2 Syntax

```
[state, z, ifail] = nag_rand_field_1d_generate(ns, s, lam, rho, state, 'm', m)
[state, z, ifail] = g05zp(ns, s, lam, rho, state, 'm', m)
```

3 Description

A one-dimensional random field $Z(x)$ in $\mathbb{R}$ is a function which is random at every point $x \in \mathbb{R}$, so $Z(x)$ is a random variable for each x. The random field has a mean function $\mu(x) = \mathbb{E}[Z(x)]$ and a symmetric non-negative definite covariance function $C(x,y) = \mathbb{E}[(Z(x) - \mu(x))(Z(y) - \mu(y))]$. $Z(x)$ is a Gaussian random field if for any choice of $n \in \mathbb{N}$ and $x_1, \ldots, x_n \in \mathbb{R}$, the random vector $[Z(x_1), \ldots, Z(x_n)]^{\mathrm{T}}$ follows a multivariate Normal distribution, which would have a mean vector $\tilde{\boldsymbol{\mu}}$ with entries $\tilde{\mu}_i = \mu(x_i)$ and a covariance matrix $\tilde{C}$ with entries $\tilde{C}_{ij} = C(x_i, x_j)$. A Gaussian random field $Z(x)$ is stationary if $\mu(x)$ is constant for all $x \in \mathbb{R}$ and $C(x,y) = C(x+a, y+a)$ for all $x, y, a \in \mathbb{R}$ and hence we can express the covariance function $C(x,y)$ as a function γ of one variable: $C(x,y) = \gamma(x-y)$. γ is known as a variogram (or more correctly, a semivariogram) and includes the multiplicative factor σ^2 representing the variance such that $\gamma(0) = \sigma^2$.

The functions nag_rand_field_1d_user_setup (g05zm) or nag_rand_field_1d_predef_setup (g05zn), along with nag_rand_field_1d_generate (g05zp), are used to simulate a one-dimensional stationary Gaussian random field, with mean function zero and variogram $\gamma(x)$, over an interval $[x_{\min}, x_{\max}]$, using an equally spaced set of N points. The problem reduces to sampling a Normal random vector $\mathbf{X}$ of size N, with mean vector zero and a symmetric Toeplitz covariance matrix A. Since A is in general expensive to factorize, a technique known as the *circulant embedding method* is used. A is embedded into a larger, symmetric circulant matrix B of size $M \geq 2(N-1)$, which can now be factorized as $B = W\Lambda W^* = R^*R$, where W is the Fourier matrix (W^* is the complex conjugate of W), Λ is the diagonal matrix containing the eigenvalues of B and $R = \Lambda^{\frac{1}{2}}W^*$. B is known as the embedding matrix. The eigenvalues can be calculated by performing a discrete Fourier transform of the first row (or column) of B and multiplying by M, and so only the first row (or column) of B is needed – the whole matrix does not need to be formed.

As long as all of the values of Λ are non-negative (i.e., B is non-negative definite), B is a covariance matrix for a random vector $\mathbf{Y}$, two samples of which can now be simulated from the real and imaginary parts of $R^*(\mathbf{U} + i\mathbf{V})$, where $\mathbf{U}$ and $\mathbf{V}$ have elements from the standard Normal distribution. Since $R^*(\mathbf{U} + i\mathbf{V}) = W\Lambda^{\frac{1}{2}}(\mathbf{U} + i\mathbf{V})$, this calculation can be done using a discrete Fourier transform of the vector $\Lambda^{\frac{1}{2}}(\mathbf{U} + i\mathbf{V})$. Two samples of the random vector $\mathbf{X}$ can now be recovered by taking the first N elements of each sample of $\mathbf{Y}$ – because the original covariance matrix A is embedded in B, $\mathbf{X}$ will have the correct distribution.

If B is not non-negative definite, larger embedding matrices B can be tried; however if the size of the matrix would have to be larger than **maxm**, an approximation procedure is used. See the documentation of nag_rand_field_1d_user_setup (g05zm) or nag_rand_field_1d_predef_setup (g05zn) for details of the approximation procedure.

nag_rand_field_1d_generate (g05zp) takes the square roots of the eigenvalues of the embedding matrix B, and its size M, as input and outputs S realizations of the random field in Z.

One of the initialization functions nag_rand_init_repeat (g05kf) (for a repeatable sequence if computed sequentially) or nag_rand_init_nonrepeat (g05kg) (for a non-repeatable sequence) must be called prior to the first call to nag_rand_field_1d_generate (g05zp).

4 References

Dietrich C R and Newsam G N (1997) Fast and exact simulation of stationary Gaussian processes through circulant embedding of the covariance matrix *SIAM J. Sci. Comput.* **18** 1088–1107

Schlather M (1999) Introduction to positive definite functions and to unconditional simulation of random fields *Technical Report ST 99–10* Lancaster University

Wood A T A and Chan G (1994) Simulation of stationary Gaussian processes in $[0,1]^d$ *Journal of Computational and Graphical Statistics* **3(4)** 409–432

5 Parameters

5.1 Compulsory Input Parameters

1: **ns** – INTEGER

The number of sample points to be generated in realizations of the random field. This must be the same value as supplied to nag_rand_field_1d_user_setup (g05zm) or nag_rand_field_1d_predef_setup (g05zn) when calculating the eigenvalues of the embedding matrix.

Constraint: **ns** ≥ 1.

2: **s** – INTEGER

S, the number of realizations of the random field to simulate.

Constraint: **s** ≥ 1.

3: **lam**(**m**) – REAL (KIND=nag_wp) array

Must contain the square roots of the eigenvalues of the embedding matrix, as returned by nag_rand_field_1d_user_setup (g05zm) or nag_rand_field_1d_predef_setup (g05zn).

Constraint: **lam**$(i) \geq 0, i = 1,2,\ldots,$ **m**.

4: **rho** – REAL (KIND=nag_wp)

Indicates the scaling of the covariance matrix, as returned by nag_rand_field_1d_user_setup (g05zm) or nag_rand_field_1d_predef_setup (g05zn).

Constraint: $0.0 <$ **rho** ≤ 1.0.

5: **state**(:) – INTEGER array

Note: the actual argument supplied **must** be the array **state** supplied to the initialization routines nag_rand_init_repeat (g05kf) or nag_rand_init_nonrepeat (g05kg).

Contains information on the selected base generator and its current state.

5.2 Optional Input Parameters

1: **m** – INTEGER

Default: the dimension of the array **lam**.

M, the size of the embedding matrix, as returned by nag_rand_field_1d_user_setup (g05zm) or nag_rand_field_1d_predef_setup (g05zn).

Constraint: $\mathbf{m} \geq \max(1, 2(\mathbf{ns} - 1))$.

5.3 Output Parameters

1: **state**(:) – INTEGER array

Contains updated information on the state of the generator.

2: $\mathbf{z}(\mathbf{ns}, \mathbf{s})$ – REAL (KIND=nag_wp) array

Contains the realizations of the random field. The jth realization, for the **ns** sample points, is stored in $\mathbf{z}(i, j)$, for $i = 1, 2, \ldots, \mathbf{ns}$. The sample points are as returned in **xx** by nag_rand_field_1d_user_setup (g05zm) or nag_rand_field_1d_predef_setup (g05zn).

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

Constraint: $\mathbf{ns} \geq 1$.

ifail $= 2$

Constraint: $\mathbf{s} \geq 1$.

ifail $= 3$

Constraint: $\mathbf{m} \geq \max(1, 2 \times (\mathbf{ns} - 1))$.

ifail $= 4$

On entry, at least one element of **lam** was negative.
Constraint: all elements of **lam** must be non-negative.

ifail $= 5$

Constraint: $0.0 \leq \mathbf{rho} \leq 1.0$.

ifail $= 6$

On entry, **state** vector has been corrupted or not initialized.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

Because samples are generated in pairs, calling this function k times, with $\mathbf{s} = s$, say, will generate a different sequence of numbers than calling the function once with $\mathbf{s} = ks$, unless s is even.

9 Example

This example calls nag_rand_field_1d_generate (g05zp) to generate 5 realizations of a random field on 8 sample points using eigenvalues calculated by nag_rand_field_1d_predef_setup (g05zn) for a symmetric stable variogram.

9.1 Program Text

```
function g05zp_example

fprintf('g05zp example results\n\n');

% Choose the Symmetric stable variogram
icov1 = nag_int(1);
params = [0.1; 1.2];

% Random Field variance
var = 0.5;
% Domain endpoints
xmin = -1;
xmax = 1;
% Number of sample points
ns = nag_int(8);
% Scaling factor, rho = 1.
icorr = nag_int(2);

% Get square roots of the eigenvalues of the embedding matrix
[lam, xx, m, approx, rho, icount, eig, ifail] = ...
  g05zn( ...
         ns, xmin, xmax, var, icov1, params, 'icorr', icorr);

fprintf('\nSize of embedding matrix = %d\n\n', m);

% Display approximation information if approximation used
if approx == 1
  fprintf('Approximation required\n\n');
  fprintf('rho = %10.5f\n', rho);
  fprintf('eig = %10.5f%10.5f%10.5f\n', eig(1:3));
  fprintf('icount = %d\n', icount);
else
  fprintf('Approximation not required\n\n');
end

state = initialize_state();

% Compute s random field realisations.
s = nag_int(5);
[state, z, ifail] = g05zp( ...
                           ns, s, lam(1:m), rho, state);

% Display realizations
fprintf('Random field realisations:\n     ');
fprintf('%10d',[1:5]);
fprintf('\n');
disp([xx, z]);

function state = initialize_state()
```

```
  genid = nag_int(1);
  subid = nag_int(1);
  seed  = [nag_int(14965)];
  [state, ifail] = g05kf( ...
                          genid, subid, seed);
```

9.2 Program Results

```
     g05zp example results

Size of embedding matrix = 16

Approximation not required

Random field realisations:
              1         2         3         4         5
   -0.8750   -0.4166   -0.8185   -0.9769    0.6741   -0.6762
   -0.6250    0.0146    1.4538    0.0248    0.5218    1.9466
   -0.3750   -0.5556    0.2913   -0.0853    0.4214   -0.1389
   -0.1250   -0.5568    0.3199   -0.6094    0.2019    0.9085
    0.1250   -0.0423    0.0486    1.4590    0.3608   -0.5288
    0.3750   -0.2806   -0.7969    0.2330    0.1335    0.4012
    0.6250    0.9298   -0.3956   -0.8455   -0.2749    0.5270
    0.8750    0.3222    1.5227   -2.1645    0.1794    1.1937
```