StochTools

Stochastic Modelling Tools

User's guide

Jean-Michel Brankart

http://pp.ige-grenoble.fr/pageperso/brankarj/

Institut des Géosciences de l'Environnement
Université Grenoble Alpes, CNRS, France

The main purpose of StochTools is to provide tools to generate random numbers, random fields, or stochastic processes of various types, to be used in stochastic modelling or data assimilation systems.

The tools are provided as a library of modules, which can be easily plugged in any existing software. This library includes:

- the generation of random numbers with various probability distribution (uniform, normal, gamma, beta, exponential, truncated exponential, truncated normal);
- the computation of the probabilty density function (pdf), the cumulative distribution function (cdf) or the inverse cdf of several probability distributions (normal, gamma, beta);
- the computation of the cdf of the product of two normal random numbers;
- the transformation of a normal random variable into a gamma or a beta random variable, and the transformation of the product of two normal random variables into a normal variable;
- the generation of random fields with specified spectrum (in 1D, in 2D or in the basis of the spherical harmonics).

1 Description of the modules

In this section, the modules are described one by one, giving for each of them: the method that has been implemented, the list of public variables and public routines (with a description of input and output data), the MPI parallelization, and an estimation of the computational cost as a function of the size of the problem.

1.1 Module: storng

The purpose of this module is to generate random numbers, according to a few specific distribution: uniform, Gaussian, gamma, and beta.

Method

The module is based on (and includes) the 64-bit KISS (Keep It Simple Stupid) random number generator distributed by George Marsaglia. KISS is based on 3 components: (1) Xorshift (XSH), period $2^{64}-1$, (2) Multiply-with-carry (MWC), period $(2^{121}+2^{63}-1)$ (3) Congruential generator (CNG), period 2^{64}. The overall period of the sequence of radom numbers is: $(2^{250}+2^{192}+2^{64}-2^{186}-2^{129})/6 \simeq 2^{247.42}$ or $10^{74.48}$.

Public routines

kiss: KISS random number generator (64-byte integers);

kiss_seed: define seeds for KISS random number generator;

kiss_save: save current state of KISS (for future restart);

kiss_load: load the saved state of KISS;

kiss_reset: reset to the default seeds;

kiss_check: check the KISS pseudo-random sequence;

kiss_uniform: real random numbers with uniform distribution in [0,1];

kiss_gaussian: real random numbers with Gaussian distribution N(0,1);

kiss_gamma: real random numbers with Gamma distribution Gamma(k,1);

kiss_beta: real random numbers with Beta distribution Beta(a,b).

Computational cost

This is a very cheap random number generator. Each call to the `kiss` function requires the following list of operations applied to 64-byte integers:

* (1), + (6), ISHFT (4), IEOR (3), .EQ. (1)

1.2 Module: stotge

The purpose of this module is to generate random numbers with truncated normal or truncated exponential distribution.

Public routines

ran_te: sample random number with truncated exponential distribution;

ran_tg: sample random number with truncated Gaussian distribution;

ranv_tg: sample random vector with truncated Gaussian distribution.

1.3 Module: stoutil

The purpose of this module is to compute the probabilty density function (pdf), the cumulative distribution function (cdf) or the inverse cdf of several probability distributions (normal, gamma, beta).

Public routines

pdf_gaussian: compute Gaussian pdf;

logpdf_gaussian: compute the logarithm of a Gaussian pdf (minus a constant);

cdf_gaussian: compute Gaussian cdf;

invcdf_gaussian: compute Gaussian inverse cdf;

pdf_gamma: compute gamma pdf;

logpdf_gamma: compute the logarithm of a gamma pdf (minus a constant);

cdf_gamma: compute gamma cdf;

invcdf_gamma: compute gamma inverse cdf;

pdf_beta: compute beta pdf;

logpdf_beta: compute the logarithm of a beta pdf (minus a constant);

cdf_beta: compute beta cdf;

invcdf_beta: compute beta inverse cdf.

1.4 Module: stogprod

The purpose of this module is to compute the cumulative distribution function (cdf) of the product of two normal random variables. This module is based on Alan Miller'implementation of the cdf (`https://jblevins.org/mirror/amiller/`).

Public routines

fnprod: compute the cdf of the product of two normal random variables.

1.5 Module: stoanam

The purpose of this module is to transform a random variable from one distribution to another.

Public routines

gau_to_gam: transform a normal random variable into a gamma random variable;

gau_to_beta: transform a normal random variable into a beta random variable;

gprod_to_gau: transform the product of two normal random variables into a normal random variable.

1.6 Module: storfg

The purpose of this module is to generate random fields with specified spectrum in the basis of the spherical harmonics. Routines to generate 1D or 2D random fields with a continuous spectrum are also included in the module, but they need to be reconsidered and they are not described below.

Method

The approach is to generate two-dimensional random fields $w(\theta, \varphi)$, function of latitude (θ) and longitude (φ), by linear combination of the sperical harmonics $Y_{lm}(\theta, \varphi)$:

$$w(\theta, \varphi) = \sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} a_{lm}\, \xi_{lm}\, Y_{lm}(\theta, \varphi) \tag{1}$$

where l and m are the degree and order of each spherical harmonics, ξ_{lm} are independent Gaussian noises (with zero mean and unit variance), and a_{lm} are spectral amplitudes defining the spatial correlation structure of the random field.

The amplitudes a_{lm} defines the spectrum of w in the basis of spherical harmonics, for instance:

$$a_{lm}^2 = \frac{k}{2l+1} \left[1 + (l/l_c)^{2p} \right]^{-1} \tag{2}$$

where the degree l_c defines the characteristic length scale ($\ell_c = R_c/l_c$, where R_c is the earth radius), the exponent p modifies the shape of the spectrum, and the coefficient k is chosen to specify the variance (σ^2) of w, i.e. so that:

$$\sum_{l=0}^{l_{\max}} \sum_{m=-l}^{l} a_{lm}^2 = \sigma^2 \tag{3}$$

Choosing the a_{lm} independent of m [as in Eq. (2)] means that the random field w is homogeneous and isotropic on the sphere.

Public routines

gen_field_2s: to generate the random field.

`ranfield` **(output)** : random field on the requested grid;

`lon` **(input)** : longitude of grid points;

`lat` **(input)** : latitude of grid points;

`pow_spect` **(input)** : callback routine providing the requested power spectrum;

`lmin` **(input)** : minimum degree of the spherical harmonics;

`lmax` **(input)** : maximum degree of the spherical harmonics.

Computational cost

The computational complexity (leading asymptotic behaviour for large systems) is given by:

$$C \sim knl_{\max}^2 = kn\left(\frac{2\pi R}{\lambda_c}\right)^2 \tag{4}$$

where n is the number of grid points in the random field, $l_{\max}$ is the maximum degree of spherical harmonics, and k is the number of operation required for one single evaulation of spherical harmonics.

2 Examples

In this section, the examples provided with the library are described, giving for each of them: the purpose of the examples, the list of modules/routines that are illustrated, the input parameters and data, the calling sequence of the library routines, with a description of inputs and outputs for each of them, and a description of the final result that is expected.

2.1 Random field on the sphere

The purpose of this example is to illustrate the generation of a random field on the sphere. Input data are: the maximum degree of the spherical harmonics used to generate the random field, the power spectrum of the random field, and the definition of the output grid (where to provide the random field). The output is a random field on the sphere written in NetCDF.