

NAG Toolbox for MATLAB**Chapter Introduction****G05 – Random Number Generators****Contents**

1	Scope of the Chapter	2
2	Background to the Problems	2
2.1	Pseudo-random Numbers	2
2.2	Quasi-random Numbers	2
2.3	Non-uniform Random Numbers	3
2.4	Copulas and Other Random Structures	3
3	Recommendations on Choice and Use of Available Functions	4
3.1	Design of the Chapter	4
3.1.1	Initialization	4
3.1.2	Repeated initialization	4
3.1.3	Copulas	4
4	Index	4
5	References	5

1 Scope of the Chapter

This chapter is concerned with the generation of sequences of independent pseudo-random and quasi-random numbers from various distributions, and the generation of pseudo-random time series from specified time series models.

2 Background to the Problems

2.1 Pseudo-random Numbers

A sequence of pseudo-random numbers is a sequence of numbers generated in some systematic way such that its statistical properties are as close as possible to those of true random numbers: for example, negligible correlation between consecutive numbers. The most common methods are based on the **multiplicative congruential** algorithm, see Knuth 1981. The basic algorithm is defined as:

$$n_i = (a \times n_{i-1}) \bmod m \quad (1)$$

The integers n_i are then divided by m to give uniformly distributed pseudo-random numbers lying in the interval $(0, 1)$.

Alternatively there is a variant known as the Wichmann–Hill algorithm, see Maclaren 1989, defined as:

$$\begin{aligned} n_{1,i} &= (a_1 \times n_{1,i-1}) \bmod m_1 \\ n_{2,i} &= (a_2 \times n_{2,i-1}) \bmod m_2 \\ n_{3,i} &= (a_3 \times n_{3,i-1}) \bmod m_3 \\ n_{4,i} &= (a_4 \times n_{4,i-1}) \bmod m_4 \\ U_i &= \left(\frac{n_{1,i}}{m_1} + \frac{n_{2,i}}{m_2} + \frac{n_{3,i}}{m_3} + \frac{n_{4,i}}{m_4} \right) \bmod 1.0 \end{aligned} \quad (2)$$

This generates pseudo-random numbers U_i , uniformly distributed in the interval $(0, 1)$.

Either of these algorithms can be selected to generate uniformly distributed pseudo-random numbers. If the basic algorithm (1) is selected then the NAG generator uses the values $a = 13^{13}$ and $m = 2^{59}$ in (1). This generator gives a **cycle length** (i.e., the number of random numbers before the sequence starts repeating itself) of 2^{57} . A good rule of thumb is never to use more numbers than the square root of the cycle length in any one experiment as the statistical properties are impaired. For closely related reasons, breaking numbers down into their bit patterns and using individual bits may cause trouble.

If the Wichmann–Hill algorithm is selected then one or more of 273 independent generators are available. Each of these is defined by the set of constants a_j and m_j for $j = 1, \dots, 4$. The constants a_j are in the range 112 to 127 and the constants m_j are prime numbers in the range 16718909 to 16776971, which are close to $2^{24} = 16777216$. These constants have been chosen so that they give good results with the spectral test, see Knuth 1981 and Maclaren 1989. The period of each Wichmann–Hill generator would be at least 2^{92} if it were not for common factors between $(m_1 - 1)$, $(m_2 - 1)$, $(m_3 - 1)$ and $(m_4 - 1)$. However, each generator should still have a period of at least 2^{80} . Further discussion of the properties of these generators is given in Maclaren 1989 where it was shown that the generated pseudo-random sequences are essentially independent of one another according to the spectral test.

The sequence given in (1) needs an initial value n_0 , known as the **seed**, while the sequence given in (2) needs four such seeds. The use of the same seed will lead to the same sequence of numbers when these are computed serially. One method of obtaining a seed is to use the real-time clock; this will give a non-repeatable sequence. It is important to note that the statistical properties of the random numbers are only guaranteed within sequences and not between sequences. Repeated initialization will thus render the numbers obtained less rather than more independent. Similarly the statistical properties of the random numbers are not guaranteed between two sequences generated using the two algorithms.

2.2 Quasi-random Numbers

Quasi-random numbers are intended primarily for use in Monte Carlo integration. Like pseudo-random numbers they are uniformly distributed but they are not statistically independent, rather they are designed to give a more even distribution in multidimensional space (uniformity). Therefore, they are often more efficient than pseudo-random numbers in multidimensional Monte Carlo methods. There are several quasi-

random generators, three of which are available in this chapter, they are the Sobol, Faure and Neiderreiter generators.

2.3 Non-uniform Random Numbers

Random numbers from other distributions may be obtained from the uniform random numbers by the use of transformations and rejection techniques, and for discrete distributions, by table based methods.

(a) Transformation Methods

For a continuous random variable, if the cumulative distribution function (CDF) is $F(x)$ then for a uniform $(0, 1)$ random variate u , $y = F^{-1}(u)$ will have CDF $F(x)$. This method is only efficient in a few simple cases such as the exponential distribution with mean μ , in which case $F^{-1}(u) = -\mu \log u$. Other transformations are based on the joint distribution of several random variables. In the bivariate case, if v and w are random variates there may be a function g such that $y = g(v, w)$ has the required distribution; for example, the Student's t -distribution with n degrees of freedom in which v has a Normal distribution, w has a gamma distribution and $g(v, w) = v\sqrt{n/w}$.

(b) Rejection Methods

Rejection techniques are based on the ability to easily generate random numbers from a distribution (called the envelope) similar to the distribution required. The value from the envelope distribution is then accepted as a random number from the required distribution with a certain probability; otherwise, it is rejected and a new number is generated from the envelope distribution.

(c) Table Search Methods

For discrete distributions, if the cumulative probabilities, $P_i = \text{Prob}(x \leq i)$, are stored in a table then, given u from a uniform $(0, 1)$ distribution, the table is searched for i such that $P_{i-1} < u \leq P_i$. The returned value i will have the required distribution. The table searching can be made faster by means of an index, see Ripley 1987. The effort required to set up the table and its index may be considerable, but the methods are very efficient when many values are needed from the same distribution.

2.4 Copulas and Other Random Structures

A copula is a function that links the univariate marginal distributions with their multivariate distribution. Sklar's theorem Sklar 1973 states that if f is an m -dimensional distribution function with continuous margins $f_1, f_2, \dots, f_m$, then f has a unique copula representation, c , such that

$$f(x_1, x_2, \dots, x_m) = c(f_1(x_1), f_2(x_2), \dots, f_m(x_m))$$

The copula, c , is a multivariate uniform distribution whose dependence structure is defined by the dependence structure of the multivariate distribution f , with

$$c(u_1, u_2, \dots, u_m) = f(f_1^{-1}(u_1), f_2^{-1}(u_2), \dots, f_m^{-1}(u_m))$$

where $u_i \in [0, 1]$. This relationship can be used to simulate variates from distributions defined by the dependence structure of one distribution and each of the marginal distributions given by another. For additional information see Nelsen 1998 or Boye Unpublished manuscript and the references therein.

In addition to random numbers from various distributions, random compound structures can be generated. These include random time series, random matrices and random samples.

The efficiency of a simulation exercise may often be increased by the use of variance reduction methods (see Morgan 1984). It is also worth considering whether a simulation is the best approach to solving the problem. For example, low-dimensional integrals are usually more efficiently calculated by functions in Chapter D01 rather than by Monte Carlo integration.

3 Recommendations on Choice and Use of Available Functions

3.1 Design of the Chapter

All the generation functions call an internal generator (selected to be either the basic generator (1) or a Wichmann–Hill generator (2)), which generates random numbers from a uniform distribution over (0,1).

3.1.1 Initialization

Prior to generating any variates the internal generator must be initialized. Two utility functions are provided for this, g05kb and g05kc, both of which allow either the basic generator or one of the 273 Wichmann–Hill generators to be chosen as the internal generator.

g05kb selects and initializes a generator to a repeatable (when executed serially) state: two calls of g05kb with the same parameter-values will result in the same subsequent sequences of random numbers (when both are generated serially). The basic generator or one of 273 Wichmann–Hill generators can be selected as the base generator.

g05kc selects and initializes a generator to a non-repeatable state, in such a way that different calls of g05kc, either in the same run or different runs of the program, will almost certainly result in different subsequent sequences of random numbers.

Functions to save and restore generator states are not required since the **iseed** parameter contains the current state and can have its values copied and stored at any time.

3.1.2 Repeated initialization

As mentioned in Section 2, it is important to note that the statistical properties of pseudo-random numbers are only guaranteed within sequences and not between sequences produced by the same generator. Repeated initialization will thus render the numbers obtained less rather than more independent. In a simple case there should be only one call to g05kb or g05kc and this call should be before any call to an actual generation function.

3.1.3 Copulas

After calling g05ra or g05rb the G01F functions in Chapter G01 can be used to convert the uniform marginal distributors into a different form as required.

4 Index

Generating samples, matrices and tables,

random correlation matrix	g05qb
random orthogonal matrix	g05qa
random permutation of an integer vector	g05na
random sample from an integer vector	g05nb
random table	g05qd

Generation of time series,

asymmetric GARCH Type II	g05h1
asymmetric GJR GARCH	g05hm
EGARCH	g05hn
symmetric GARCH or asymmetric GARCH Type I	g05hk

univariate ARMA model,

Normal errors	g05pa
vector ARMA model,	
Normal errors	g05pc

Pseudo-random numbers,

array of variates from multivariate distributions,

multinomial distribution	g05mr
Normal distribution	g05lx
Student's <i>t</i> distribution	g05ly

Copulas	
Gaussian Copula	g05ra
Student's <i>t</i> Copula	g05rb
initialize generator,	
nonrepeatable sequence	g05kc
repeatable sequence	g05kb
single variate from multivariate distributions,	
Normal distribution	g05lz
single variate, from a univariate distribution	
logical value true or false	g05ke
real number from the continuous uniform distribution	g05ka
vector of variates from discrete univariate distributions,	
binomial distribution	g05mj
geometric distribution	g05mb
hypergeometric distribution	g05ml
logarithmic distribution	g05md
negative binomial distribution	g05mc
Poisson distribution	g05mk
uniform distribution	g05ma
user-supplied distribution	g05mz
variate array from discrete distributions with array of parameters,	
Poisson distribution with varying mean	g05me
vectors of variates from continuous univariate distributions,	
beta distribution	g05le
Cauchy distribution	g05ll
chi-square distribution	g05lc
exponential mix distribution	g05lq
<i>F</i> -distribution	g05ld
gamma distribution	g05lf
logistic distribution	g05ln
lognormal distribution	g05lk
negative exponential distribution	g05lj
Normal distribution	g05la
Student's <i>t</i> -distribution	g05lb
triangular distribution	g05lh
uniform distribution	g05lg
von Mises distribution	g05lp
Weibull distribution	g05lm
Quasi-random numbers,	
array of variates from univariate distributions,	
Faure generator	g05yd
Log normal distribution	g05yk
Neiderreiter generator	g05yh
Normal distribution	g05yj
Sobol generator	g05yf
initialize generator,	
Faure generators	g05yc
Neiderreiter generators	g05yg
Sobol generators	g05ye

5 References

- Boye E Unpublished manuscript Copulas for Finance: A reading guide and some applications Financial Econometrics Research Centre, City University Business School, London
- Knuth D E 1981 *The Art of Computer Programming (Volume 2)* (2nd Edition) Addison–Wesley
- Maclaren N M 1989 The generation of multiple independent sequences of pseudorandom numbers *Appl. Statist.* **38** 351–359

Morgan B J T 1984 *Elements of Simulation* Chapman and Hall

Nelsen R B 1998 *An Introduction to Copulas. Lecture Notes in Statistics 139* Springer

Ripley B D 1987 *Stochastic Simulation* Wiley

Sklar A 1973 Random Variables: Joint Distribution Functions and Copulas *Kybernetika* **9** 499–460
