Chapter G05

Random Number Generators

Contents

1 Scope of the Chapter

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

2 Background to the Problems

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 method used is a **multiplicative congruential** algorithm defined as:

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

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

The NAG generator uses the values $a = 13^{13}$ and $m = 2^{59}$; for further details see G05CAF. 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.

The sequence given in (1) needs an initial value n_0, known as the **seed**. The use of the same seed will lead to the same sequence of numbers. One method of obtaining the 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.

Random numbers from other distributions may be obtained from the uniform random numbers by the use of transformations, rejection techniques, and for discrete distributions 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 [4]. 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.

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 [3]). 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 routines in Chapter D01 rather than by Monte Carlo integration.

3 Recommendations on Choice and Use of Available Routines

Note. Refer to the Users' Note for your implementation to check that a routine is available.

3.1 Design of the Chapter

All the generation routines call – directly or indirectly – an internal basic generator, which generates random numbers from a uniform distribution over (0,1). Thus a call to any generation routine will affect all subsequent random numbers produced by any other routine in the chapter. Despite this effect, the values will remain as independent as if the different sequences were produced separately.

Two utility routines are provided to initialize the basic generator:

G05CBF initializes it to a repeatable state, dependent on an integer parameter: two calls of G05CBF with the same parameter-value will result in the same subsequent sequences of random numbers.

G05CCF initializes it to a non-repeatable state, in such a way that different calls of G05CCF, either in the same run or different runs of the program, will almost certainly result in different subsequent sequences of random numbers.

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. Repeated initialization will thus render the numbers obtained less rather than more independent. In a simple case there should be only one call to G05CBF or G05CCF, which should be before any call to an actual generation routine.

Two other utility routines, G05CFF and G05CGF, are provided to save or restore the state of the basic generator (including the seed of the multiplicative congruential method used by the basic generator). G05CFF and G05CGF can be used to produce two or more sequences of numbers simultaneously, where some are repeatable and some are not; for example, this can be used to simulate signal and noise. As their overheads are not negligible, numbers should be produced in batches when this technique is used. While they can be used to save the state of the internal generator between jobs, the two arrays must be restored accurately. The corresponding process between machines, while sometimes possible, is not advised.

3.2 Selection of Routine

For three of the commonest continuous distributions – uniform, exponential, and Normal – there is a choice between calling a function to return a single random number and calling a subroutine to fill an array with a sequence of random numbers; the latter is likely to be much more efficient on vector-processing machines.

Distribution	**Function returning a single number**	**Subroutine returning an array of numbers**
uniform over (0,1)	G05CAF	G05FAF
uniform over (a, b)	G05DAF	G05FAF
exponential	G05DBF	G05FBF
Normal	G05DDF	G05FDF

For two discrete distributions, the uniform and Poisson, there is a choice between routines that use indexed search tables, which are suitable for the generation of many variates from the distribution with the same parameters, and routines that are more efficient in the single call situation when the parameters may be changing.

Distribution	**Single call**	**Set up table**
discrete uniform	G05DYF	G05EBF
Poisson	G05DRF	G05ECF

G05EBF and G05ECF return a reference array which is then used by G05EYF.

The following distributions are also available. Those indicated can return more than one value per call.

(a) Continuous Distributions

Beta distribution (multiple)	`G05FEF`
Cauchy distribution	`G05DFF`
Chi-square distribution	`G05DHF`
F-distribution	`G05DKF`
Gamma distribution (multiple)	`G05FFF`
Logistic distribution	`G05DCF`
Lognormal distribution	`G05DEF`
Student's t-distribution	`G05DJF`
von Mises distribution	`G05FSF`
Weibull distribution	`G05DPF`

(b) Multivariate Distributions

Multivariate Normal distribution	`G05EAF G05EZF`

(c) Discrete Distributions using table search

Binomial distribution	`G05EDF`
Hypergeometric distribution	`G05EFF`
Negative binomial distribution	`G05EEF`
User-supplied distribution	`G05EXF`

The above routines set up the table and index in a reference array; G05EYF can then be called to generate the random variate from the information in the reference array.

(d) Generation of Time Series

Univariate ARMA model, Normal errors	`G05EGF G05EWF`
Vector ARMA model, Normal errors	`G05HDF`

(e) Sampling and Permutation

Random permutation of an integer vector	`G05EHF`
Random sample from an integer vector	`G05EJF`
Random logical value	`G05DZF`

(f) Random Matrices

Random orthogonal matrix	`G05GAF`
Random correlation matrix	`G05GBF`

3.3 Programming Advice

Take care when programming calls to those routines in this chapter which are functions. The reason is that different calls with the same parameters are intended to give different results.

For example, if you wish to assign to Z the difference between two successive random numbers generated by G05CAF, beware of writing

```
Z = G05CAF(X) - G05CAF(X)
```

It is quite legitimate for a Fortran compiler to compile zero, one or two calls to G05CAF; if two calls, they may be in either order (if zero or one calls are compiled, Z would be set to zero). A safe method to program this would be

```
X = G05CAF(X)
Y = G05CAF(Y)
Z = X-Y
```

Another problem that can occur is that an optimising compiler may move a call to a function out of a loop. Thus, the same value would be used for all iterations of the loop, instead of a different random number being generated at each iteration. If this problem occurs, consult an expert on your Fortran compiler.

All the routines in this chapter rely on information stored in common blocks, which must be saved between calls. This need not be a matter of concern unless a program is split into overlays; in such a case, the safest course is to ensure that the G05 routines are in the root overlay.

4 Routines Withdrawn or Scheduled for Withdrawal

Since Mark 13 the following routines have been withdrawn. Advice on replacing calls to these routines is given in the document 'Advice on Replacement Calls for Withdrawn/Superseded Routines'.

G05DGF G05DLF G05DMF

5 References

[1] Dagpunar J (1988) *Principles of Random Variate Generation* Oxford University Press

[2] Knuth D E (1981) *The Art of Computer Programming (Volume 2)* Addison–Wesley (2nd Edition)

[3] Morgan B J T (1984) *Elements of Simulation* Chapman and Hall

[4] Ripley B D (1987) *Stochastic Simulation* Wiley