NAG Toolbox

nag_rand_init_nonrepeat (g05kg)

1 Purpose

nag_rand_init_nonrepeat (g05kg) initializes the selected base generator to generate a non-repeatable sequence of variates. The base generator can then be used by the group of pseudorandom number functions (see nag_rand_init_leapfrog (g05kh)–nag_rand_init_skipahead (g05kj), nag_rand_permute (g05nc), nag_rand_sample (g05nd), nag_rand_times_garch_asym1 (g05pd)–nag_rand_times_mv_varma (g05pj), nag_rand_matrix_orthog (g05px)–nag_rand_matrix_2waytable (g05pz), nag_rand_copula_students_t (g05rc), nag_rand_copula_normal (g05rd), nag_rand_multivar_students_t (g05ry), nag_rand_multivar_normal (g05rz) and nag_rand_dist_uniform01 (g05sa)–nag_rand_int_uniform (g05tl)) and the quasi-random scrambled sequence initialization function, nag_rand_quasi_init_scrambled (g05yn).

2 Syntax

```
[state, ifail] = nag_rand_init_nonrepeat(genid, subid)
[state, ifail] = g05kg(genid, subid)
```

3 Description

nag_rand_init_nonrepeat (g05kg) selects a base generator through the input value of the arguments **genid** and **subid**, and then initializes it based on the values taken from the real-time clock, resulting in the same base generator yielding different sequences of random numbers each time the calling program is run. It should be noted that there is no guarantee of statistical properties between sequences, only within sequences.

A definition of some of the terms used in this description, along with details of the various base generators can be found in the G05 Chapter Introduction.

4 References

L'Ecuyer P and Simard R (2002) *TestU01: a software library in ANSI C for empirical testing of random number generators* Departement d'Informatique et de Recherche Operationnelle, Universite de Montreal http://www.iro.umontreal.ca/~lecuyer

Maclaren N M (1989) The generation of multiple independent sequences of pseudorandom numbers *Appl. Statist.* **38** 351–359

Matsumoto M and Nishimura T (1998) Mersenne twister: a 623-dimensionally equidistributed uniform pseudorandom number generator *ACM Transactions on Modelling and Computer Simulations*

Wichmann B A and Hill I D (2006) Generating good pseudo-random numbers *Computational Statistics and Data Analysis* **51** 1614–1622

Wikramaratna R S (1989) ACORN - a new method for generating sequences of uniformly distributed pseudo-random numbers *Journal of Computational Physics* **83** 16–31

5 Parameters

5.1 Compulsory Input Parameters

1: **genid** – INTEGER

Must contain the type of base generator to use.

genid = 1
NAG basic generator.

genid $= 2$
Wichmann Hill I generator.

genid $= 3$
Mersenne Twister.

genid $= 4$
Wichmann Hill II generator.

genid $= 5$
ACORN generator.

genid $= 6$
L'Ecuyer MRG32k3a generator.

See the G05 Chapter Introduction for details of each of the base generators.

Constraint: **genid** $= 1$, 2, 3, 4, 5 or 6.

2: **subid** – INTEGER

If **genid** $= 2$, **subid** indicates which of the 273 sub-generators to use. In this case, the $((|\mathbf{subid}| + 272)\ mod\ 273) + 1$ sub-generator is used.

If **genid** $= 5$, **subid** indicates the values of k and p to use, where k is the order of the generator, and p controls the size of the modulus, M, with $M = 2^{(p \times 30)}$. If **subid** < 1, the default values of $k = 10$ and $p = 2$ are used, otherwise values for k and p are calculated from the formula, **subid** $= k + 1000(p - 1)$.

If **genid** $= 6$ and **subid** mod $2 = 0$ the range of the generator is set to $(0, 1]$, otherwise the range is set to $(0, 1)$; in this case the sequence is identical to the implementation of MRG32k3a in TestU01 (see L'Ecuyer and Simard (2002)) for identical seeds.

For all other values of **genid**, **subid** is not referenced.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **state**$(lstate)$ – INTEGER array

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

2: **ifail** – INTEGER

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

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of ifail on exit.**

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

Constraint: **genid** $= 1$, 2, 3, 4, 5 or 6.

ifail $= 4$

Constraint: $lstate \leq 0$ or .

ifail $= -1$

Required length of **state** array returned in *lstate* but **state** array 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

None.

9 Example

This example prints the first five pseudorandom real numbers from a uniform distribution between 0 and 1, generated by nag_rand_dist_uniform01 (g05sa) after initialization by nag_rand_init_nonrepeat (g05kg).

9.1 Program Text

```
function g05kg_example

fprintf('g05kg example results\n\n');

% genid and subid identify the base generator
genid = nag_int(1);
subid =  nag_int(1);

% Initialize the generator to an unrepeatable sequence
[state, ifail] = g05kg(genid, subid);

% The number of pseudorandom numbers to be generated
n = nag_int(5);

% Generate the variates
[state, x, ifail] = g05sa( ...
                          n, state);

% Display variates
disp(x);
```

9.2 Program Results

```
g05kg example results

    0.3331
    0.5686
    0.7844
    0.5503
    0.1498
```