NAG Toolbox

nag_rand_init_leapfrog (g05kh)

1 Purpose

nag_rand_init_leapfrog (g05kh) allows for the generation of multiple, independent, sequences of pseudorandom numbers using the leap-frog method.

2 Syntax

```
[state, ifail] = nag_rand_init_leapfrog(n, k, state)
[state, ifail] = g05kh(n, k, state)
```

3 Description

nag_rand_init_leapfrog (g05kh) adjusts a base generator to allow multiple, independent, sequences of pseudorandom numbers to be generated via the leap-frog method (see the G05 Chapter Introduction for details).

If, prior to calling nag_rand_init_leapfrog (g05kh) the base generator defined by **state** would produce random numbers $x_1, x_2, x_3, \ldots$, then after calling nag_rand_init_leapfrog (g05kh) the generator will produce random numbers $x_k, x_{k+n}, x_{k+2n}, x_{k+3n}, \ldots$.

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_init_leapfrog (g05kh).

The leap-frog algorithm can be used in conjunction with the NAG basic generator, both the Wichmann–Hill I and Wichmann–Hill II generators, the Mersenne Twister and L'Ecuyer.

4 References

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

5 Parameters

5.1 Compulsory Input Parameters

1: **n** – INTEGER

n, the total number of sequences required.

Constraint: $\mathbf{n} > 0$.

2: **k** – INTEGER

k, the number of the current sequence.

Constraint: $0 < \mathbf{k} \leq \mathbf{n}$.

3: **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

None.

5.3 Output Parameters

1: **state**(:) – INTEGER array

Contains updated information on the state of the generator.

2: **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: **n** ≥ 1.

ifail $= 2$

Constraint: $0 <$ **k** $\leq$ **n**.

ifail $= 3$

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

ifail $= 4$

On entry, cannot use leap-frog with the base generator defined by **state**.

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

The leap-frog method tends to be less efficient than other methods of producing multiple, independent sequences. See the G05 Chapter Introduction for alternative choices.

9 Example

This example creates three independent sequences using nag_rand_init_leapfrog (g05kh), after initialization by nag_rand_init_repeat (g05kf). Five variates from a uniform distribution are then generated from each sequence using nag_rand_dist_uniform01 (g05sa).

9.1 Program Text

```
     function g05kh_example

fprintf('g05kh example results\n\n');

% Initialize the seed
seed = [nag_int(1762543)];

% genid and subid identify the base generator
genid = nag_int(1);
subid =  nag_int(1);
lstate = nag_int(17);
lseed =  nag_int(1);

% n is the number of streams
n = nag_int(3);
% nv is the number of variates
nv = nag_int(5);
% Hold state and variates in successive columns
state = zeros(lstate, n, nag_int_name);
x = zeros(nv, n);

% Initialize the generator to a repeatable sequence for streams
[state1, ifail] = g05kf( ...
                         genid, subid, seed);
% Prepare n streams
for i=1:n
  % Prepare stream i
  state(:, i) = state1;
  [state(:, i), ifail] = g05kh( ...
                                n, i, state(:,i));
end

% Generate nv variates from a uniform distribution, from each stream
for i=1:n
  [state(:, i), x(:, i), ifail] = g05sa( ...
                                         nv, state(:, i));
end

% Display variates by stream
for i=1:n
  fprintf(' Stream %d\n', i);
  disp(x(:,i));
end
```

9.2 Program Results

```
     g05kh example results

 Stream 1
    0.7460
    0.4925
    0.4982
    0.2580
    0.5938

 Stream 2
    0.7983
    0.3843
    0.6717
    0.6238
    0.2785

 Stream 3
```

```
    0.1046
    0.7871
    0.0505
    0.0535
    0.2375
```
