NAG Library Function Document

nag_rand_skip_ahead_power2 (g05kkc)

1 Purpose

nag_rand_skip_ahead_power2 (g05kkc) allows for the generation of multiple, independent, sequences of pseudorandom numbers using the skip-ahead method. The base pseudorandom number sequence defined by **state** is advanced 2^n places.

2 Specification

```
#include <nag.h>
#include <nagg05.h>
void nag_rand_skip_ahead_power2 (Integer n, Integer state[], NagError *fail)
```

3 Description

nag_rand_skip_ahead_power2 (g05kkc) adjusts a base generator to allow multiple, independent, sequences of pseudorandom numbers to be generated via the skip-ahead method (see the g05 Chapter Introduction for details).

If, prior to calling nag_rand_skip_ahead_power2 (g05kkc) the base generator defined by **state** would produce random numbers $x_1, x_2, x_3, \ldots$, then after calling nag_rand_skip_ahead_power2 (g05kkc) the generator will produce random numbers $x_{2^n+1}, x_{2^n+2}, x_{2^n+3}, \ldots$.

One of the initialization functions nag_rand_init_repeatable (g05kfc) (for a repeatable sequence if computed sequentially) or nag_rand_init_nonrepeatable (g05kgc) (for a non-repeatable sequence) must be called prior to the first call to nag_rand_skip_ahead_power2 (g05kkc).

The skip-ahead algorithm can be used in conjunction with any of the six base generators discussed in the g05 Chapter Introduction.

4 References

Haramoto H, Matsumoto M, Nishimura T, Panneton F and L'Ecuyer P (2008) Efficient jump ahead for F2-linear random number generators *INFORMS J. on Computing* **20(3)** 385–390

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

5 Arguments

1: **n** – Integer *Input*

On entry: n, where the number of places to skip-ahead is defined as 2^n.

Constraint: **n** ≥ 0.

2: **state**$[dim]$ – Integer *Communication Array*

Note: the dimension, dim, of this array is dictated by the requirements of associated functions that must have been previously called. This array MUST be the same array passed as argument **state** in the previous call to nag_rand_init_repeatable (g05kfc) or nag_rand_init_nonrepeatable (g05kgc).

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

3: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_ARRAY_SIZE

On entry, the **state** vector defined on initialization is not large enough to perform a skip-ahead (applies to Mersenne Twister base generator). See the initialization function nag_rand_init_repeatable (g05kfc) or nag_rand_init_nonrepeatable (g05kgc).

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

NE_INT_ARRAY

On entry, cannot use skip-ahead with the base generator defined by **state**.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in the Essential Introduction for further information.

NE_INVALID_STATE

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

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

Calling nag_rand_skip_ahead_power2 (g05kkc) and then generating a series of uniform values using nag_rand_basic (g05sac) is equivalent to, but more efficient than, calling nag_rand_basic (g05sac) and discarding the first 2^n values. This may not be the case for distributions other than the uniform, as some distributional generators require more than one uniform variate to generate a single draw from the required distribution.

10 Example

This example initializes a base generator using nag_rand_init_repeatable (g05kfc) and then uses nag_rand_skip_ahead_power2 (g05kkc) to advance the sequence 2^{17} places before generating five variates from a uniform distribution using nag_rand_basic (g05sac).

10.1 Program Text

```
/* nag_rand_skip_ahead_power2 (g05kkc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>

#define LSEED 1

int main(void)
{
  /* Scalars */
  Integer exit_status = 0;
  Integer lstate, n, nv, subid, i;
  /* Arrays */
  Integer seed[LSEED];
  Integer *state = 0;
  double *x = 0;
  char cgenid[40];
  /* Nag types */
  Nag_BaseRNG genid;
  NagError fail;

  /* Initialise the error structure */
  INIT_FAIL(fail);

  printf("nag_rand_skip_ahead_power2 (g05kkc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Read in the base generator information and seed */
#ifdef _WIN32
   scanf_s("%39s%"NAG_IFMT "%"NAG_IFMT "%*[^\n] ",cgenid, _countof(cgenid),&su-
bid,&seed[0]);
#else
  scanf("%39s%"NAG_IFMT "%"NAG_IFMT "%*[^\n] ",cgenid,&subid,&seed[0]);
#endif
  genid = (Nag_BaseRNG) nag_enum_name_to_value(cgenid);

  /* Query to get the require length of state array */
  lstate = -1;
  nag_rand_init_repeatable(genid, subid, seed, LSEED, state, &lstate, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Allocate state */
  if (!(state = NAG_ALLOC((lstate), Integer)))
    {
```

```
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Initialize the generator to a repeatable sequence */
  nag_rand_init_repeatable(genid, subid, seed, LSEED, state, &lstate, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n", fail.message);
    exit_status = 2;
    goto END;
  }

  /* Read in the skip ahead and sample size */
#ifdef _WIN32
  scanf_s("%"NAG_IFMT "%"NAG_IFMT "%*[^\n] ",&n,&nv);
#else
  scanf("%"NAG_IFMT "%"NAG_IFMT "%*[^\n] ",&n,&nv);
#endif
  if (!(x = NAG_ALLOC((nv), double)))
    {
      printf("Allocation failure\n");
      exit_status = -2;
      goto END;
    }

  /* Advance the sequence 2**n places */
  nag_rand_skip_ahead_power2(n, state, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_skip_ahead_power2 (g05kkc).\n%s\n",
            fail.message);
    exit_status = 3;
    goto END;
  }

  /* Generate NV variates from a uniform distribution */
  nag_rand_basic(nv, state, x, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_basic (g05sac).\n%s\n", fail.message);
    exit_status = 4;
    goto END;
  }

  /* Display the variates */
  for (i = 0; i < nv; i++) printf("%10.4f\n", x[i]);
  printf("\n");

 END:
  NAG_FREE(x);
  NAG_FREE(state);
  return exit_status;
}
```

10.2 Program Data

```
nag_rand_skip_ahead_power2 (g05kkc) Example Program Data
Nag_MersenneTwister  1  6344398     :: genid, subid, seed[0]
17  5                               :: n, nv
```

10.3 Program Results

```
nag_rand_skip_ahead_power2 (g05kkc) Example Program Results

    0.0152
    0.0904
    0.4607
    0.8997
    0.5923
```