NAG Library Function Document

nag_rand_discrete_uniform (g05tlc)

1 Purpose

nag_rand_discrete_uniform (g05tlc) generates a vector of pseudorandom integers uniformly distributed over the interval $[a, b]$.

2 Specification

```
#include <nag.h>
#include <nagg05.h>
```

```
void nag_rand_discrete_uniform (Integer n, Integer a, Integer b,
     Integer state[], Integer x[], NagError *fail)
```

3 Description

nag_rand_discrete_uniform (g05tlc) generates the next n values y_i from a uniform $(0, 1]$ generator (see nag_rand_basic (g05sac) for details) and applies the transformation

$$x_i = a + \lfloor (b - a + 1) y_i \rfloor,$$

where $\lfloor z \rfloor$ is the integer part of the real value z. The function ensures that the values x_i lie in the closed interval $[a, b]$.

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_discrete_uniform (g05tlc).

4 References

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

5 Arguments

1: **n** – Integer *Input*

On entry: n, the number of pseudorandom numbers to be generated.

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

2: **a** – Integer *Input*

3: **b** – Integer *Input*

On entry: the end points a and b of the uniform distribution.

Constraint: $\mathbf{a} \leq \mathbf{b}$.

4: **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.

5: **x**[**n**] – Integer *Output*

On exit: the n pseudorandom numbers from the specified uniform distribution.

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

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

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument $\langle\mathit{value}\rangle$ had an illegal value.

NE_INT

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

NE_INT_2

On entry, $\mathbf{a} = \langle\mathit{value}\rangle$ and $\mathbf{b} = \langle\mathit{value}\rangle$.
Constraint: $\mathbf{b} \geq \mathbf{a}$.

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.

NE_INVALID_STATE

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

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

This example prints five pseudorandom integers from a discrete uniform distribution between -5 and 5, generated by a single call to nag_rand_discrete_uniform (g05tlc), after initialization by nag_rand_init_repeatable (g05kfc).

10.1 Program Text

```
/* nag_rand_discrete_uniform (g05tlc) Example Program.
 *
 * Copyright 2008, Numerical Algorithms Group.
 *
 * Mark 9, 2009.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
```

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

int main(void)
{
  /* Integer scalar and array declarations */
  Integer     exit_status = 0;
  Integer     i, lstate;
  Integer     *state = 0, *x = 0;

  /* NAG structures */
  NagError    fail;

  /* Set the distribution parameters */
  Integer     a = -5;
  Integer     b = 5;

  /* Set the sample size */
  Integer     n = 5;

  /* Choose the base generator */
  Nag_BaseRNG genid = Nag_Basic;
  Integer     subid = 0;

  /* Set the seed */
  Integer     seed[] = { 1762543 };
  Integer     lseed = 1;

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

  printf(
          "nag_rand_discrete_uniform (g05tlc) Example Program Results\n\n");

  /* Get the length of the 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 arrays */
  if (!(state = NAG_ALLOC(lstate, Integer)) ||
      !(x = NAG_ALLOC(n, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Initialise 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 = 1;
      goto END;
    }

  /* Generate the variates*/
  nag_rand_discrete_uniform(n, a, b, state, x, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rand_discrete_uniform (g05tlc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Display the variates*/
  for (i = 0; i < n; i++)
    printf("%12ld\n", x[i]);

 END:
  NAG_FREE(state);
  NAG_FREE(x);

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_rand_discrete_uniform (g05tlc) Example Program Results

           2
          -4
           3
           3
          -4
```