NAG C Library Function Document

nag_rngs_beta (g05lec)

1 Purpose

nag_rngs_beta (g05lec) generates a vector of pseudo-random numbers taken from a beta distribution with a and b.

2 Specification

```
void nag_rngs_beta (double a, double b, Integer n, double x[], Integer igen,
     Integer iseed[], NagError *fail)
```

3 Description

The beta distribution has PDF (probability density function)

$$f(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\, x^{a-1}(1-x)^{b-1} \quad \text{if } 0 \le x \le 1;\ a, b > 0.0,$$

$$f(x) = 0 \quad \text{otherwise.}$$

One of four algorithms is used to generate the variates depending on the values of a and b. Let α be the maximum and β be the minimum of a and b. Then the algorithms are as follows:

(i) If $\alpha < 0.5$, Jhnk's algorithm is used, see for example Dagpunar (1988). This generates the beta variate as $u_1^{1/a} / \left(u_1^{1/a} + u_2^{1/b} \right)$, where u_1 and u_2 are uniformly distributed random variates;

(ii) If $\beta > 1$, the algorithm BB given by Cheng (1978) is used. This involves the generation of an observation from a beta distribution of the second kind by the envelope rejection method using a log-logistic target distribution and then transforming it to a beta variate;

(iii) If $\alpha > 1$ and $\beta < 1$, the switching algorithm given by Atkinson (1979) is used. The two target distributions used are $f_1(x) = \beta x^{\beta}$ and $f_2(x) = \alpha(1-x)^{\beta-1}$, along with the approximation to the switching parameter of $t = (1-\beta)/(\alpha+1-\beta)$;

(iv) In all other cases, Cheng's BC algorithm (see Cheng (1978)) is used with modifications suggested by Dagpunar (1988). This algorithm is similar to BB, used when $\beta > 1$, but is tuned for small values of a and b.

One of the initialisation functions nag_rngs_init_repeatable (g05kbc) (for a repeatable sequence if computed sequentially) or nag_rngs_init_nonrepeatable (g05kcc) (for a non-repeatable sequence) must be called prior to the first call to nag_rngs_beta (g05lec).

4 References

Atkinson A C (1979) A family of switching algorithms for the computer generation of beta random variates *Biometrika* **66** 141–5

Cheng R C H (1978) Generating beta variates with nonintegral shape parameters *Comm. ACM* **21** 317–322

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

Hastings N A J and Peacock J B (1975) *Statistical Distributions* Butterworths

5 Parameters

1: **a** – double *Input*

On entry: the parameter, a, of the beta distribution.

Constraint: **a** > 0.0.

2: **b** – double *Input*

On entry: the parameter, b, of the beta distribution.

Constraint: **b** > 0.0.

3: **n** – Integer *Input*

On entry: the number, n, of pseudo-random numbers to be generated.

Constraint: **n** ≥ 0.

4: **x**$[dim]$ – double *Output*

Note: the dimension, dim, of the array **x** must be at least $\max(1, \mathbf{n})$.

On exit: the n pseudo-random numbers from the specified beta distribution.

5: **igen** – Integer *Input*

On entry: must contain the identification number for the generator to be used to return a pseudo-random number and should remain unchanged following initialisation by a prior call to one of the functions nag_rngs_init_repeatable (g05kbc) or nag_rngs_init_nonrepeatable (g05kcc).

6: **iseed**$[4]$ – Integer *Input/Output*

On entry: contains values which define the current state of the selected generator.

On exit: contains updated values defining the new state of the selected generator.

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

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 0.

NE_REAL

On entry, **b** $= \langle value \rangle$.
Constraint: **b** > 0.0.

On entry, **a** $= \langle value \rangle$.
Constraint: **a** > 0.0.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ had an illegal value.

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 consult NAG for assistance.

7 Accuracy

Not applicable.

8 Further Comments

To generate an observation, y, from the beta distribution of the second kind from an observation, x, generated by nag_rngs_beta (g05lec) the transformation, $y = x/(1-x)$, may be used.

9 Example

The example program prints a set of five pseudo-random numbers from a beta distribution with parameters $a = 2.0$ and $b = 2.0$, generated by a single call to nag_rngs_beta (g05lec), after initialisation by nag_rngs_init_repeatable (g05kbc).

9.1 Program Text

```
/* nag_rngs_beta(g05lec) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

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

int main(void)
{
  /* Scalars */
  Integer  igen, j, n;
  Integer  exit_status=0;
  NagError fail;

  /* Arrays */
  double   *x=0;
  Integer  iseed[4];

  INIT_FAIL(fail);
  Vprintf("g05lec Example Program Results\n\n");

  n = 5;
  /* Allocate memory */
  if ( !(x = NAG_ALLOC(n, double)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Initialise the seed to a repeatable sequence */
  iseed[0] = 1762543;
  iseed[1] = 9324783;
  iseed[2] = 42344;
  iseed[3] = 742355;
  /* igen identifies the stream. */
  igen = 1;
  g05kbc(&igen, iseed);

  Vprintf("Beta Dist --- A=2.0, B=2.0\n");

  g05lec(2.0, 2.0, n, x, igen, iseed, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from g05lec.\n%s\n", fail.message);
```

```
      exit_status = 1;
      goto END;
    }
  for (j = 0; j < n; ++j)
    {
      Vprintf("%10.4f\n",   x[j]);
    }
 END:
  if (x) NAG_FREE(x);
  return exit_status;
}
```

9.2 Program Data

None.

9.3 Program Results

```
g05lec Example Program Results

Beta Dist --- A=2.0, B=2.0
    0.4334
    0.8888
    0.5604
    0.3799
    0.5064
```