NAG C Library Function Document

nag_rngs_neg_bin (g05mcc)

1 Purpose

nag_rngs_neg_bin (g05mcc) generates a vector of pseudo-random integers from the discrete negative binomial distribution with parameter m and probability p of success at a trial.

2 Specification

```
void nag_rngs_neg_bin (Integer mode, Integer m, double p, Integer n, Integer x[],
     Integer igen, Integer iseed[], double r[], NagError *fail)
```

3 Description

nag_rngs_neg_bin (g05mcc) generates n integers x_i from a discrete negative binomial distribution, where the probability of $x_i = I$ (I successes before m failures) is

$$P(x_i = I) = \frac{(m+I-1)!}{I!(m-1)!} \times p^I \times (1-p)^m, \quad I = 0, 1, \ldots.$$

The variates can be generated with or without using a search table and index. If a search table is used then it is stored with the index in a reference vector and subsequent calls to nag_rngs_neg_bin (g05mcc) with the same parameter value can then use this reference vector to generate further variates.

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_neg_bin (g05mcc).

4 References

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

5 Parameters

1: **mode** – Integer *Input*

On entry: a code for selecting the operation to be performed by the function:

mode $= 0$

Set up reference vector only.

mode $= 1$

Generate variates using reference vector set up in a prior call to nag_rngs_neg_bin (g05mcc).

mode $= 2$

Set up reference vector and generate variates.

mode $= 3$

Generate variates without using the reference vector.

Constraint: $0 \leq$ **mode** ≤ 3.

2: **m** – Integer *Input*

On entry: the number of failures, m, of the distribution.

Constraint: **m** ≥ 0.

3: **p** – double *Input*

On entry: the parameter p of the negative binomial distribution representing the probability of success at a single trial.

Constraint: $0.0 \leq \mathbf{p} < 1.0$.

4: **n** – Integer *Input*

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

Constraint: $\mathbf{n} \geq 1$.

5: **x**[**n**] – Integer *Output*

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

6: **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).

7: **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.

8: **r**[dim] – double *Input/Output*

Note: the dimension, dim, of the array **r** must be at least $20 + (20\sqrt{\mathbf{m} \times \mathbf{p}} + 30 \times \mathbf{p})/(1 - \mathbf{p})$ when **mode** < 3 and at least 1 otherwise.

On exit: the reference vector.

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

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **mode** = ⟨*value*⟩.
Constraint: $0 \leq \mathbf{mode} \leq 3$.

On entry, **m** = ⟨*value*⟩.
Constraint: $\mathbf{m} \geq 0$.

On entry, **n** = ⟨*value*⟩.
Constraint: $\mathbf{n} \geq 1$.

NE_DIM_INFEASIBLE

p and **m** are such that the reference vector length would exceed integer range. We recommend setting **mode** = 3.

NE_PREV_CALL

p or **m** is not the same as when **r** was set up in a previous call. Previous value of **p** = ⟨*value*⟩, **p** = ⟨*value*⟩. Previous value of **m** = ⟨*value*⟩, **m** = ⟨*value*⟩.

NE_REAL

On entry, $\mathbf{p} < 0.0$ or $\mathbf{p} \geq 1.0$: **p** = ⟨*value*⟩.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ 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

None.

9 Example

The example program prints five pseudo-random integers from a negative binomial distribution with parameters $m = 60$ and $p = 0.999$, generated by a single call to nag_rngs_neg_bin (g05mcc), after initialisation by nag_rngs_init_repeatable (g05kbc).

9.1 Program Text

```
/* nag_rngs_neg_bin(g05mcc) 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 */
  double   p;
  Integer  i, igen, m, n, nr;
  Integer  exit_status=0;
  NagError fail;

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

  INIT_FAIL(fail);
  Vprintf("g05mcc Example Program Results\n\n");
  nr = 1;
  n = 20;

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

  /* Set the distribution parameter P */
  p = 0.999;
  m = 60;
  /* 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);

  /* Choose MODE = 3 because P close to 1 */
  g05mcc(3, m, p, n, x, igen, iseed, r, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from g05mcc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  for (i = 0; i < n; ++i)
    {
      Vprintf("%12ld", x[i]);

      Vprintf("\n");
    }
 END:
  if (r) NAG_FREE(r);
  if (x) NAG_FREE(x);
  return exit_status;
}
```

9.2 Program Data

None.

9.3 Program Results

```
g05mcc Example Program Results

       49822
       73300
       57798
       64791
       73197
       58394
       61279
       57579
       56346
       70705
       59782
       58087
       61213
       81333
       47030
       57082
       67922
       56991
       46611
       62691
```