NAG Library Function Document

nag_rand_geom (g05tcc)

1 Purpose

nag_rand_geom (g05tcc) generates a vector of pseudorandom integers from the discrete geometric distribution with probability p of success at a trial.

2 Specification

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

```
void nag_rand_geom (Nag_ModeRNG mode, Integer n, double p, double r[],
     Integer lr, Integer state[], Integer x[], NagError *fail)
```

3 Description

nag_rand_geom (g05tcc) generates n integers x_i from a discrete geometric distribution, where the probability of $x_i = I$ (a first success after $I + 1$ trials) is

$$P(x_i = I) = p \times (1 - p)^I, \quad I = 0, 1, \ldots.$$

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_rand_geom (g05tcc) with the same parameter value can then use this reference vector to generate further variates. If the search table is not used (as recommended for small values of p) then a direct transformation of uniform variates is used.

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_geom (g05tcc).

4 References

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

5 Arguments

1: **mode** – Nag_ModeRNG *Input*

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

mode = Nag_InitializeReference
Set up reference vector only.

mode = Nag_GenerateFromReference
Generate variates using reference vector set up in a prior call to nag_rand_geom (g05tcc).

mode = Nag_InitializeAndGenerate
Set up reference vector and generate variates.

mode = Nag_GenerateWithoutReference
Generate variates without using the reference vector.

Constraint: **mode** = Nag_InitializeReference, Nag_GenerateFromReference, Nag_InitializeAndGenerate or Nag_GenerateWithoutReference.

2: **n** – Integer *Input*

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

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

3: **p** – double *Input*

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

Constraint: ***machine precision*** $\leq \mathbf{p} \leq 1.0$ (see nag_machine_precision (X02AJC)).

4: **r**[**lr**] – double *Communication Array*

On entry: if **mode** = Nag_GenerateFromReference, the reference vector from the previous call to nag_rand_geom (g05tcc).

If **mode** = Nag_GenerateWithoutReference, **r** is not referenced and may be **NULL**.

On exit: if **mode** ≠ Nag_GenerateWithoutReference, the reference vector.

5: **lr** – Integer *Input*

On entry: the dimension of the array **r**.

Suggested value:

if **mode** ≠ Nag_GenerateWithoutReference, $\mathbf{lr} = 8 + 42/\mathbf{p}$ approximately (see Section 9); otherwise $\mathbf{lr} = 1$.

Constraints:

if **mode** = Nag_InitializeReference or Nag_InitializeAndGenerate, $\mathbf{lr} \geq 30/\mathbf{p} + 8$;
if **mode** = Nag_GenerateFromReference, **lr** should remain unchanged from the previous call to nag_rand_geom (g05tcc).

6: **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.

7: **x**[**n**] – Integer *Output*

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

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

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

NE_INT

On entry, **lr** is too small when **mode** = Nag_InitializeReference or Nag_InitializeAndGenerate: **lr** = $\langle\mathit{value}\rangle$, minimum length required = $\langle\mathit{value}\rangle$.

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

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 How to Use the NAG Library and its Documentation 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 How to Use the NAG Library and its Documentation for further information.

NE_PREV_CALL

p is not the same as when **r** was set up in a previous call.
Previous value of **p** = $\langle\mathit{value}\rangle$ and **p** = $\langle\mathit{value}\rangle$.

NE_REAL

On entry, **p** = $\langle\mathit{value}\rangle$.
Constraint: ***machine precision*** $\leq$ **p** ≤ 1.0.

p is so small that **lr** would have to be larger than the largest representable integer. Use **mode** = Nag_GenerateWithoutReference instead. **p** = $\langle\mathit{value}\rangle$

NE_REF_VEC

On entry, some of the elements of the array **r** have been corrupted or have not been initialized.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_rand_geom (g05tcc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

The time taken to set up the reference vector, if used, increases with the length of array **r**. However, if the reference vector is used, the time taken to generate numbers decreases as the space allotted to the index part of **r** increases. Nevertheless, there is a point, depending on the distribution, where this improvement becomes very small and the suggested value for the length of array **r** is designed to approximate this point.

If **p** is very small then the storage requirements for the reference vector and the time taken to set up the reference vector becomes prohibitive. In this case it is recommended that the reference vector is not used. This is achieved by selecting **mode** = Nag_GenerateWithoutReference.

10 Example

This example prints 10 pseudorandom integers from a geometric distribution with parameter $p = 0.001$, generated by a single call to nag_rand_geom (g05tcc), after initialization by nag_rand_init_repeatable (g05kfc).

10.1 Program Text

```
/* nag_rand_geom (g05tcc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* 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, lr, lstate;
  Integer *state = 0, *x = 0;

  /* NAG structures */
  NagError fail;

  /* Double scalar and array declarations */
  double *r = 0;

  /* Set the distribution parameters */
  double p = 0.0010e0;

  /* Set the mode we will be using. As p is small
     we will not use a reference vector */
  Nag_ModeRNG mode = Nag_GenerateWithoutReference;

  /* Set the sample size */
  Integer n = 10;

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

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

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

  printf("nag_rand_geom (g05tcc) 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;
  }

  /* Calculate the size of the reference vector, if any */
  lr = (mode != Nag_GenerateWithoutReference) ? 8 + 42 / p : 0;

  /* Allocate arrays */
  if (!(r = NAG_ALLOC(lr, double)) ||
      !(state = NAG_ALLOC(lstate, Integer)) || !(x = NAG_ALLOC(n, 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 = 1;
    goto END;
  }

  /* Generate the variates, dont use a reference vector as p is close to 0 */
  nag_rand_geom(mode, n, p, r, lr, state, x, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_geom (g05tcc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

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

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

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_rand_geom (g05tcc) Example Program Results

         451
        2238
         292
         225
        2256
         708
         955
         239
         696
         397
```