

# NAG Library Function Document

## nag\_rand\_hypergeometric (g05tec)

### 1 Purpose

nag\_rand\_hypergeometric (g05tec) generates a vector of pseudorandom integers from the discrete hypergeometric distribution of the number of specified items in a sample of size  $l$ , taken from a population of size  $k$  with  $m$  specified items in it.

### 2 Specification

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

void nag_rand_hypergeometric (Nag_ModeRNG mode, Integer n, Integer ns,
    Integer np, Integer m, double r[], Integer lr, Integer state[],
    Integer x[], NagError *fail)
```

### 3 Description

nag\_rand\_hypergeometric (g05tec) generates  $n$  integers  $x_i$  from a discrete hypergeometric distribution, where the probability of  $x_i = I$  is

$$P(i = I) = \frac{l!m!(k-l)!(k-m)!}{I!(l-I)!(m-I)!(k-m-l+I)!k!} \quad \text{if } I = \max(0, m+l-k), \dots, \min(l, m),$$

$$P(i = I) = 0 \quad \text{otherwise.}$$

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\_hypergeometric (g05tec) with the same parameter values can then use this reference vector to generate further variates. The reference array is generated by a recurrence relation if  $lm(k-l)(k-m) < 50k^3$ , otherwise Stirling's approximation 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\_hypergeometric (g05tec).

### 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\_hypergeometric (g05tec).

**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:*  $n \geq 0$ .
- 3: **ns** – Integer *Input*  
*On entry:*  $l$ , the sample size of the hypergeometric distribution.  
*Constraint:*  $0 \leq ns \leq np$ .
- 4: **np** – Integer *Input*  
*On entry:*  $k$ , the population size of the hypergeometric distribution.  
*Constraint:*  $np \geq 0$ .
- 5: **m** – Integer *Input*  
*On entry:*  $m$ , the number of specified items of the hypergeometric distribution.  
*Constraint:*  $0 \leq m \leq np$ .
- 6: **r[*lr*]** – double *Communication Array*  
*On entry:* if **mode** = Nag\_GenerateFromReference, the reference vector from the previous call to nag\_rand\_hypergeometric (g05tec).  
 If **mode** = Nag\_GenerateWithoutReference, **r** is not referenced and may be NULL.  
*On exit:* if **mode**  $\neq$  Nag\_GenerateWithoutReference, the reference vector.
- 7: **lr** – Integer *Input*  
*On entry:* the dimension of the array **r**.  
*Suggested value:*  
 if **mode**  $\neq$  Nag\_GenerateWithoutReference,  
 $lr = 28 + 20 \times \sqrt{(ns \times m \times (np - m) \times (np - ns)) / np^3}$  approximately;  
 otherwise **lr** = 1.  
*Constraints:*  
 if **mode** = Nag\_InitializeReference or Nag\_InitializeAndGenerate, **lr** must not be too small, but the limit is too complicated to specify;  
 if **mode** = Nag\_GenerateFromReference, **lr** must remain unchanged from the previous call to nag\_rand\_hypergeometric (g05tec).
- 8: **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.

- 9: **x[n]** – Integer *Output*  
*On exit:* the pseudorandom numbers from the specified hypergeometric distribution.
- 10: **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\_BAD\_PARAM

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

### NE\_INT

On entry, **lr** is too small when **mode** = Nag\_InitializeReference or Nag\_InitializeAndGenerate:  
**lr** =  $\langle value \rangle$ , minimum length required =  $\langle value \rangle$ .

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

On entry, **np** =  $\langle value \rangle$ .  
 Constraint: **np**  $\geq$  0.

### NE\_INT\_2

On entry, **m** =  $\langle value \rangle$  and **np** =  $\langle value \rangle$ .  
 Constraint:  $0 \leq \mathbf{m} \leq \mathbf{np}$ .

On entry, **ns** =  $\langle value \rangle$  and **np** =  $\langle value \rangle$ .  
 Constraint:  $0 \leq \mathbf{ns} \leq \mathbf{np}$ .

### 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.

### NE\_PREV\_CALL

The value of **ns**, **np** or **m** is not the same as when **r** was set up in a previous call with  
**mode** = Nag\_InitializeReference or Nag\_InitializeAndGenerate.

### 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\_hypergeometric (g05tec) 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' Note for your implementation for any additional implementation-specific information.

## 9 Further Comments

None.

## 10 Example

The example program prints 20 pseudorandom integers from a hypergeometric distribution with  $l = 500$ ,  $m = 900$  and  $n = 1000$ , generated by a single call to nag\_rand\_hypergeometric (g05tec), after initialization by nag\_rand\_init\_repeatable (g05kfc).

### 10.1 Program Text

```

/* nag_rand_hypergeometric (g05tec) Example Program.
 *
 * Copyright 2014 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    lr, i, lstate;
    Integer    *state = 0, *x = 0;

    /* NAG structures */
    NagError   fail;
    Nag_ModeRNG mode;

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

    /* Set the distribution parameters */
    Integer    ns = 500;
    Integer    m = 900;
    Integer    np = 1000;

    /* Set the sample size */
    Integer    n = 20;

    /* 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_hypergeometric (g05tec) Example Program Results\n\n");

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

/* Calculate the size of the reference vector */
lr = 28 + 20 *
    (int) sqrt(((double) m * ((double) ns / (double) np) *
        ((double)(np - m) / (double) np) *
        ((double)(np - ns) / (double) np)));

/* 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;
}

/* Initialise the generator to a repeatable sequence */
nag_rand_init_repeatabe(genid, subid, seed, lseed, state, &lstate, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_init_repeatabe (g05kfc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}

/* Generate the variates, initialising the reference
vector at the same time */
mode = Nag_InitializeAndGenerate;
nag_rand_hypergeometric(mode, n, ns, np, m, r, lr, state, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_hypergeometric (g05tec).\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\_hypergeometric (g05tec) Example Program Results

```
452
444
453
454
444
450
449
454
450
452
442
447
451
442
451
447
447
462
456
450
```

---