

# NAG Library Function Document

## nag\_rand\_logical (g05tbc)

### 1 Purpose

nag\_rand\_logical (g05tbc) generates a vector of pseudorandom logical values – Nag\_TRUE with probability  $p$  and Nag\_FALSE with probability  $(1 - p)$ .

### 2 Specification

```
#include <nag.h>
#include <nagg05.h>
void nag_rand_logical (Integer n, double p, Integer state[], Nag_Boolean x[],
                      NagError *fail)
```

### 3 Description

nag\_rand\_logical (g05tbc) generates  $n$  logical values  $x_i$  from the relation

$$y_i < p$$

where  $y_i$  is a pseudorandom number from a uniform distribution over  $(0, 1]$ , generated by nag\_rand\_basic (g05sac) using the values of **state** as input to this function.

One of the initialization functions nag\_rand\_init\_repeatable (g05kfc) (for a repeatable sequence if computed sequentially) or nag\_rand\_init\_nonrepeatable (g05kge) (for a non-repeatable sequence) must be called prior to the first call to nag\_rand\_logical (g05tbc).

### 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 logical values to be generated.  
*Constraint:*  $n \geq 0$ .
- 2: **p** – double *Input*  
*On entry:* must contain the probability of nag\_rand\_logical (g05tbc) returning Nag\_TRUE.  
*Constraint:*  $0.0 \leq p \leq 1.0$ .
- 3: **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 (g05kge).  
*On entry:* contains information on the selected base generator and its current state.  
*On exit:* contains updated information on the state of the generator.

4: **x[n]** – Nag\_Boolean *Output*  
*On exit:* the  $n$  logical values.

5: **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  $\langle value \rangle$  had an illegal value.

### NE\_INT

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

### 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\_REAL

On entry,  $\mathbf{p} = \langle value \rangle$ .  
 Constraint:  $0.0 \leq \mathbf{p} \leq 1.0$ .

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

nag\_rand\_logical (g05tbc) 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

This example prints the first 20 pseudorandom logical values generated by `nag_rand_logical` (g05tbc) after initialization by `nag_rand_init_repeatabl` (g05kfc), when the probability of a `Nag_TRUE` value is 0.5.

### 10.1 Program Text

```

/* nag_rand_logical (g05tbc) 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)
{
    /*Logical scalar and array declarations */
    Nag_Boolean *x = 0;
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer i, lstate;
    Integer *state = 0;

    /* NAG structures */
    NagError fail;

    /* Set the distribution parameters */
    double p = 0.5e0;

    /* 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;

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

    printf("nag_rand_logical (g05tbc) Example Program Results\n\n");

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

```

```

/* Allocate arrays */
if (!(state = NAG_ALLOC(lstate, Integer)) ||
    !(x = NAG_ALLOC(n, Nag_Boolean)))
{
    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 */
nag_rand_logical(n, p, state, x, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_logical (g05tbc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Display the variates */
for (i = 0; i < n; i++)
    printf("%c\n", (x[i]) ? 'T' : 'F');

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

    return exit_status;
}

```

## 10.2 Program Data

None.

## 10.3 Program Results

nag\_rand\_logical (g05tbc) Example Program Results

```

F
T
F
F
T
T
T
F
T
F
T
T
F
T
F
T
T
F
F

```

---