# NAG Library Function Document

# nag_rand_chi_sq (g05sdc)

## 1 Purpose

nag_rand_chi_sq (g05sdc) generates a vector of pseudorandom numbers taken from a $\chi^2$-distribution with $\nu$ degrees of freedom.

## 2 Specification

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

## 3 Description

The distribution has PDF (probability density function)

$$f(x) = \frac{x^{\nu/2-1} \times e^{-x/2}}{2^{\nu/2} \times (\nu/2-1)!} \quad \text{if } x > 0;$$

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

This is the same as a gamma distribution with parameters $\nu/2$ and 2.

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_chi_sq (g05sdc).

## 4 References

Kendall M G and Stuart A (1969) *The Advanced Theory of Statistics (Volume 1)* (3rd Edition) Griffin

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 numbers to be generated.

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

2: **df** – Integer *Input*

*On entry*: $\nu$, the number of degrees of freedom of the distribution.

*Constraint*: $\mathbf{df} \geq 1$.

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 (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.

4: **x**[**n**] – double *Output*

*On exit*: the $n$ pseudorandom numbers from the specified $\chi^2$-distribution.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

**NE_BAD_PARAM**

On entry, argument ⟨*value*⟩ had an illegal value.

**NE_INT**

On entry, **df** = ⟨*value*⟩.
Constraint: **df** $\geq 1$.

On entry, **n** = ⟨*value*⟩.
Constraint: **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.

**NE_INVALID_STATE**

On entry, **state** vector has been corrupted or not initialized.

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

nag_rand_chi_sq (g05sdc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

## 9 Further Comments

The time taken by nag_rand_chi_sq (g05sdc) increases with $\nu$.

## 10 Example

This example prints five pseudorandom numbers from a $\chi^2$-distribution with five degrees of freedom, generated by a single call to nag_rand_chi_sq (g05sdc), after initialization by nag_rand_init_repeatable (g05kfc).

### 10.1 Program Text

```
/* nag_rand_chi_sq (g05sdc) Example Program.
 *
 * Copyright 2008, 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     i, lstate;
  Integer     *state = 0;

  /* NAG structures */
  NagError    fail;

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

  /* Set the distribution parameters */
  Integer     df = 5;

  /* Set the sample size */
  Integer     n = 5;

  /* 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_chi_sq (g05sdc) 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;
    }

  /* Allocate arrays */
  if (!(x = NAG_ALLOC(n, double)) ||
      !(state = NAG_ALLOC(lstate, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Initialise 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_chi_sq(n, df, state, x, &fail);
  if (fail.code != NE_NOERROR)
```

```
    {
      printf("Error from nag_rand_chi_sq (g05sdc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Display the variates*/
  for (i = 0; i < n; i++)
    printf("%10.4f\n", x[i]);

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

  return exit_status;
}
```

## 10.2 Program Data

None.

## 10.3 Program Results

```
nag_rand_chi_sq (g05sdc) Example Program Results

    4.4731
    5.9371
    1.7636
    2.9812
    4.3280
```