

# NAG Library Function Document

## nag\_rand\_von\_mises (g05src)

### 1 Purpose

nag\_rand\_von\_mises (g05src) generates a vector of pseudorandom numbers from a von Mises distribution with concentration parameter  $\kappa$ .

### 2 Specification

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

### 3 Description

The von Mises distribution is a symmetric distribution used in the analysis of circular data. The PDF (probability density function) of this distribution on the circle with mean direction  $\mu_0 = 0$  and concentration parameter  $\kappa$ , can be written as:

$$f(\theta) = \frac{e^{\kappa \cos \theta}}{2\pi I_0(\kappa)},$$

where  $\theta$  is reduced modulo  $2\pi$  so that  $-\pi \leq \theta < \pi$  and  $\kappa \geq 0$ . For very small  $\kappa$  the distribution is almost the uniform distribution, whereas for  $\kappa \rightarrow \infty$  all the probability is concentrated at one point.

The  $n$  variates,  $\theta_1, \theta_2, \dots, \theta_n$ , are generated using an envelope rejection method with a wrapped Cauchy target distribution as proposed by Best and Fisher (1979) and described by Dagpunar (1988).

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\_von\_mises (g05src).

### 4 References

Best D J and Fisher N I (1979) Efficient simulation of the von Mises distribution *Appl. Statist.* **28** 152–157

Dagpunar J (1988) *Principles of Random Variate Generation* Oxford University Press

Mardia K V (1972) *Statistics of Directional Data* Academic Press

### 5 Arguments

1: **n** – Integer *Input*

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

*Constraint:*  $n \geq 0$ .

2: **vk** – double *Input*

*On entry:*  $\kappa$ , the concentration parameter of the required von Mises distribution.

*Constraint:*  $0.0 < \mathbf{vk} \leq \sqrt{\text{nag\_real\_largest\_number}/2.0}$ .

- 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** $[n]$  – double *Output*  
*On exit:* the  $n$  pseudorandom numbers from the specified von Mises distribution.
- 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 2.3.1.2 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,  $n = \langle value \rangle$ .

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.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 2.7.6 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 2.7.5 in How to Use the NAG Library and its Documentation for further information.

### NE\_REAL

On entry,  $vk \leq 0.0$  or  $vk$  too large:  $vk = \langle value \rangle$ .

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

nag\_rand\_von\_mises (g05src) 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

For a given number of random variates the generation time increases slightly with increasing  $\kappa$ .

## 10 Example

This example prints the first five pseudorandom numbers from a von Mises distribution with  $\kappa = 1.0$ , generated by a single call to nag\_rand\_von\_mises (g05src), after initialization by nag\_rand\_init\_repea table (g05kfc).

### 10.1 Program Text

```

/* nag_rand_von_mises (g05src) 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, lstate;
    Integer *state = 0;

    /* NAG structures */
    NagError fail;

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

    /* Set the distribution parameters */
    double vk = 1.0e0;

    /* 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;

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

    printf("nag_rand_von_mises (g05src) 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;
}

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

/* Initialize 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 */
nag_rand_von_mises(n, vk, state, x, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_von_mises (g05src).\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\_von\_mises (g05src) Example Program Results

```

1.2947
-1.9542
-0.6464
-1.4172
1.2536

```

---