NAG C Library Function Document

nag_rngs_von_mises (g05lpc)

1 Purpose

nag_rngs_von_mises (g05lpc) generates a vector of pseudo-random numbers from a von Mises distribution with concentration parameter κ.

2 Specification

```
void nag_rngs_von_mises (double vk, Integer n, double x[], Integer igen,
     Integer iseed[], NagError *fail)
```

3 Description

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

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

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

The n variates, $\theta_1, \theta_2, \ldots, \theta_n$, 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 initialisation functions nag_rngs_init_repeatable (g05kbc) (for a repeatable sequence if computed sequentially) or nag_rngs_init_nonrepeatable (g05kcc) (for a non-repeatable sequence) must be called prior to the first call to nag_rngs_von_mises (g05lpc).

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 Parameters

1: **vk** – double *Input*

On entry: the concentration parameter, κ, of the required von Mises distribution.

Constraint: **vk** > 0.0.

2: **n** – Integer *Input*

On entry: the number, n, of pseudo-random numbers to be generated.

Constraint: **n** ≥ 0.

3: **x**$[dim]$ – double *Output*

Note: the dimension, dim, of the array **x** must be at least $\max(1, \mathbf{n})$.

On exit: the n pseudo-random numbers from the specified von Mises distribution.

4: **igen** – Integer *Input*

On entry: must contain the identification number for the generator to be used to return a pseudo-random number and should remain unchanged following initialisation by a prior call to one of the functions nag_rngs_init_repeatable (g05kbc) or nag_rngs_init_nonrepeatable (g05kcc).

5: **iseed**[4] – Integer *Input/Output*

On entry: contains values which define the current state of the selected generator.

On exit: contains updated values defining the new state of the selected generator.

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

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

NE_REAL

On entry, **vk** $= \langle value \rangle$.
Constraint: **vk** > 0.0.

NE_BAD_PARAM

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

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 consult NAG for assistance.

7 Accuracy

Not applicable.

8 Further Comments

For a given number of random variates the generation time increases slightly with increasing κ.

If **vk** is supplied too large (i.e., **vk** $> \sqrt{(}$nag_real_largest_number (X02ALC)())) then floating point overflow will occur in internal calculation.

9 Example

The example program prints the first five pseudo-random real numbers from a von Mises distribution with $\kappa = 1.0$, generated by a single call to nag_rngs_von_mises (g05lpc), after initialisation by nag_rngs_init_repeatable (g05kbc).

9.1 Program Text

```
/* nag_rngs_von_mises(g05lpc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */
```

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

int main(void)
{
  /* Scalars */
  Integer  igen, j, m;
  Integer  exit_status=0;
  NagError fail;

  /* Arrays */
  double   *x=0;
  Integer  iseed[4];

  INIT_FAIL(fail);
  Vprintf("g05lpc Example Program Results\n\n");

  m = 5;
  /* Allocate memory */
  if ( !(x = NAG_ALLOC(m, double)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Initialise the seed to a repeatable sequence */
  iseed[0] = 1762543;
  iseed[1] = 9324783;
  iseed[2] = 42344;
  iseed[3] = 742355;
  /* igen identifies the stream. */
  igen = 1;
  g05kbc(&igen, iseed);

  g05lpc(1.0, m, x, igen, iseed, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from g05lpc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  for (j = 0; j < m; ++j)
    {
      Vprintf("%10.4f\n",   x[j]);
    }
 END:
  if (x) NAG_FREE(x);
  return exit_status;
}
```

9.2 Program Data

None.

9.3 Program Results

```
g05lpc Example Program Results

   -1.1339
   -2.5880
   -0.6178
    0.0519
   -0.9584
```