NAG C Library Function Document

nag_rngs_copula_normal (g05rac)

1 Purpose

nag_rngs_copula_normal (g05rac) sets up a reference vector and generates an array of pseudo-random numbers from a Normal (Gaussian) copula with covariance matrix C.

2 Specification

```
#include <nag.h>
#include <nagg05.h>
void nag_rngs_copula_normal (Nag_OrderType order, Integer mode, Integer m,
     const double c[], Integer pdc, Integer n, double x[], Integer pdx,
     Integer igen, Integer iseed[], double r[], Integer lr, NagError *fail)
```

3 Description

The Gaussian copula, c, is defined by

$$c(u_1, u_2, \cdots, u_m; C) = \Phi_C\left(\phi_{C_{11}}^{-1}(u_1), \phi_{C_{22}}^{-1}(u_2), \cdots, \phi_{C_{mm}}^{-1}(u_m)\right)$$

where m is the number of dimensions, Φ_C is the multivariate Normal density function with mean zero and covariance matrix C and $\phi_{C_{ii}}^{-1}$ is the inverse of the univariate Normal density function with mean zero and variance C_{ii}.

Routine nag_rgsn_matrix_multi_normal (g05lyc) is used to generate a vector from a multivariate Normal distribution and function nag_prob_normal (g01eac) is used to convert each element of that vector into a uniformly distributed value between zero and one.

One of the initialization 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_copula_normal (g05rac).

4 References

Nelsen R B (1998) *An Introduction to Copulas. Lecture Notes in Statistics 139* Springer

Sklar A (1973) Random Variables: Joint Distribution Functions and Copulas *Kybernetika* **9** 499–460

5 Arguments

1: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = **Nag_RowMajor**. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this argument.

Constraint: **order** = **Nag_RowMajor** or **Nag_ColMajor**.

2: **mode** – Integer *Input*

On entry: selects the operation to be performed:

mode = 0

Initialize and generate random numbers.

mode $= 1$

Initialize only (i.e., set up reference vector).

mode $= 2$

Generate random numbers using previously set up reference vector.

Constraint: $0 \leq$ **mode** ≤ 2.

3: **m** – Integer *Input*

On entry: m, the number of dimensions of the distribution.

Constraint: **m** > 0.

4: **c**$[dim]$ – const double *Input*

Note: the dimension, *dim*, of the array **c** must be at least **pdc** $\times$ **m**.

If **order** $=$ **Nag_ColMajor**, the (i,j)th element of the matrix C is stored in **c**$[(j-1) \times$ **pdc** $+ i - 1]$.

If **order** $=$ **Nag_RowMajor**, the (i,j)th element of the matrix C is stored in **c**$[(i-1) \times$ **pdc** $+ j - 1]$.

On entry: the covariance matrix of the distribution. Only the upper triangle need be set.

Constraint: **c** must be positive semi-definite to ***machine precision***

5: **pdc** – Integer *Input*

On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **c**.

Constraint: **pdc** $\geq$ **m**.

6: **n** – Integer *Input*

On entry: n, the number of random variates required.

Constraint: **n** ≥ 1.

7: **x**$[dim]$ – double *Output*

Note: the dimension, *dim*, of the array **x** must be at least

$\max(1,$ **pdx** $\times$ **m**$)$ when **order** $=$ **Nag_ColMajor**;
$\max(1,$ **n** $\times$ **pdx**$)$ when **order** $=$ **Nag_RowMajor**.

If **order** $=$ **Nag_ColMajor**, the (i,j)th element of the matrix X is stored in **x**$[(j-1) \times$ **pdx** $+ i - 1]$.

If **order** $=$ **Nag_RowMajor**, the (i,j)th element of the matrix X is stored in **x**$[(i-1) \times$ **pdx** $+ j - 1]$.

On exit: the array of pseudo-random multivariate Normal vectors generated by the function.

8: **pdx** – Integer *Input*

On entry: the stride separating matrix row or column elements (depending on the value of **order**) in the array **x**.

Constraints:

if **order** $=$ **Nag_ColMajor**, **pdx** $\geq$ **n**;
if **order** $=$ **Nag_RowMajor**, **pdx** $\geq$ **m**.

9: **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 initialization by a prior call to one of the functions nag_rngs_init_repeatable (g05kbc) or nag_rngs_init_nonrepeatable (g05kcc).

10: **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.

11: **r**[**lr**] – double *Input/Output*

On entry: if **mode** = 2, the reference vector as set up by nag_rngs_copula_normal (g05rac) in a previous call with **mode** = 0 or 1.

On exit: if **mode** = 0 or 1, the reference vector that can be used in subsequent calls to nag_rngs_copula_normal (g05rac) with **mode** = 2.

12: **lr** – Integer *Input*

On entry: the dimension of the array **r** as declared in the function from which nag_rngs_copula_normal (g05rac) is called. If **mode** = 2, it must be the same as the value of **lr** specified in the prior call to nag_rngs_copula_normal (g05rac) with **mode** = 0 or 1.

Constraint: **lr** $>$ **m**(**m** + 1).

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

The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

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

NE_INT

On entry, invalid value for **igen**. Ensure **igen** has not changed since random number generator was initialized.

On entry, **m** = ⟨*value*⟩.
Constraint: **m** $>$ 0.

On entry, **mode** $<$ 0 or **mode** $>$ 2: **mode** = ⟨*value*⟩.

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

On entry, **pdc** = ⟨*value*⟩.
Constraint: **pdc** $>$ 0.

On entry, **pdx** = ⟨*value*⟩.
Constraint: **pdx** $>$ 0.

NE_INT_2

m is not the same as when **r** was set up in a previous call. Previous value = ⟨*value*⟩, **m** = ⟨*value*⟩.

On entry, **lr** $<$ **m** $\times$ **m** + 1: **lr** = ⟨*value*⟩, (**m** $\times$ **m** + 1) = ⟨*value*⟩.

On entry, **pdc** = ⟨*value*⟩, **m** = ⟨*value*⟩.
Constraint: **pdc** $\geq$ **m**.

On entry, **pdx** = ⟨*value*⟩, **m** = ⟨*value*⟩.
Constraint: **pdx** $\geq$ **m**.

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.

NE_POS_DEF

The covariance matrix **c** is not positive semi-definite to ***machine precision***.

7 Accuracy

See Section 7 of the document for nag_rgsn_matrix_multi_normal (g05lyc) for an indication of the accuracy of the underlying multivariate Normal distribution.

8 Further Comments

None.

9 Example

The example program prints ten pseudo-random observations from a Normal copula with covariance matrix

$$\begin{bmatrix} 1.69 & 0.39 & -1.86 & 0.07 \\ 0.39 & 98.01 & -7.07 & -0.71 \\ -1.86 & -7.07 & 11.56 & 0.03 \\ 0.07 & -0.71 & 0.03 & 0.01 \end{bmatrix},$$

generated by nag_rngs_copula_normal (g05rac). All ten observations are generated by a single call to nag_rngs_copula_normal (g05rac) with **mode** $= 0$. The random number generator is initialized by nag_rngs_init_repeatable (g05kbc).

9.1 Program Text

```
/* nag_rngs_copula_normal (g05rac) Example Program.
 *
 * Copyright 2004 Numerical Algorithms Group.
 *
 * Mark 8, 2004.
 */

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

int main(void)
{
  /* Scalars */
  Integer exit_status, i, igen, j, lr, m, mode, n, pdc, pdx;

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

  /* Nag types */
  NagError fail;
  Nag_OrderType order;

#ifdef NAG_COLUMN_MAJOR
#define C(I, J) c[(J-1)*pdc + I - 1]
#define X(I, J) x[(J-1)*pdx + I - 1]
  order = Nag_ColMajor;
#else
#define C(I, J) c[(I-1)*pdc + J - 1]
```

```
#define X(I, J) x[(I-1)*pdx + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);
  exit_status = 0;

  /* Set the number of variables and variates */
  m = 4;
  n = 10;
  lr = m * (m + 1) + 1;
  pdc = m;
  pdx = m;

  /* Allocate memory */
  if ( !(c = NAG_ALLOC(pdc * pdc, double)) ||
       !(r = NAG_ALLOC(lr, double)) ||
       !(x = NAG_ALLOC(pdx * n, double)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  Vprintf("nag_rngs_copula_normal (g05rac) Example Program Results\n\n");

  /* Initialise the seed to a repeatable sequence */
  iseed[0] = 1762543;
  iseed[1] = 9324783;
  iseed[2] = 42344;
  iseed[3] = 742355;

  /* Choose the random generator to use */
  igen = 1;

  /* Initialise the random generator */
  /* nag_rngs_init_repeatable (g05kbc).
   * Initialize seeds of a given generator for random number
   * generating functions (that pass seeds explicitly) to give
   * a repeatable sequence
   */
  nag_rngs_init_repeatable(&igen, iseed);

  /* Input the upper triangle portion of the covariance matrix */
  C(1, 1) = 1.69;
  C(1, 2) = 0.39;
  C(1, 3) = -1.86;
  C(1, 4) = 0.07;
  C(2, 2) = 98.01;
  C(2, 3) = -7.07;
  C(2, 4) = -0.71;
  C(3, 3) = 11.56;
  C(3, 4) = 0.03;
  C(4, 4) = 0.01;

  /* Set the mode */
  mode = 0;

  /* Set up reference vector and generate n numbers */
  /* nag_rngs_copula_normal (g05rac).
   * Generates a matrix of random numbers from a Gaussian
   * Copula, seeds and generator passed explicitly
   */
  nag_rngs_copula_normal(order, mode, m, c, pdc, n, x, pdx, igen, iseed, r, lr,
                         &fail);

  /* Display the results */
  for (i=1; i<=n; ++i)
    {
      for (j=1; j<=m; ++j)
        {
```

```
          Vprintf("%10.4f ", X(i,j));
        }
      Vprintf("\n");
    }

 END:
  if (c) NAG_FREE(c);
  if (r) NAG_FREE(r);
  if (x) NAG_FREE(x);

  return exit_status;
}
```

9.2 Program Data

None.

9.3 Program Results

```
nag_rngs_copula_normal (g05rac) Example Program Results

    0.9819     0.1689     0.0712     0.9428
    0.2525     0.7025     0.5261     0.1134
    0.4771     0.8504     0.8684     0.4064
    0.0900     0.4690     0.7162     0.2592
    0.3835     0.0400     0.8975     0.7915
    0.5492     0.9685     0.4275     0.1749
    0.2064     0.3430     0.9595     0.5140
    0.7369     0.6728     0.0191     0.3639
    0.8970     0.0732     0.5617     0.9151
    0.2198     0.0157     0.8901     0.8911
```