NAG C Library Function Document

nag_rngs_multi_normal (g05lzc)

1 Purpose

nag_rngs_multi_normal (g05lzc) sets up a reference vector and generates a vector of pseudo-random numbers from a multivariate Normal distribution with mean vector a and covariance matrix C.

2 Specification

```
void nag_rngs_multi_normal (Nag_OrderType order, Integer mode, Integer n,
     const double xmu[], const double c[], Integer ic, double x[], Integer igen,
     Integer iseed[], double r[], NagError *fail)
```

3 Description

When the covariance matrix is non-singular (i.e., strictly positive-definite), the distribution has probability density function

$$f(x) = \sqrt{\frac{|C^{-1}|}{(2\pi)^n}} \exp\left\{-(x-a)^T C^{-1}(x-a)\right\}$$

where n is the number of dimensions, C is the covariance matrix, a is the vector of means and x is the vector of positions.

Covariance matrices are symmetric and positive semi-definite. Given such a matrix C, there exists a lower triangular matrix L such that $LL^T = C$. L is not unique, if C is singular.

nag_rngs_multi_normal (g05lzc) decomposes C to find such an L. It then stores n, a and L in the reference vector r which is used to generate a vector x of independent standard Normal pseudo-random numbers. It then returns the vector $a + Lx$, which has the required multivariate Normal distribution.

It should be noted that this function will work with a singular covariance matrix C, provided C is positive semi-definite, despite the fact that the above formula for the probability density function is not valid in that case. Wilkinson (1965) should be consulted if further information is required.

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_multi_normal (g05lzc).

4 References

Knuth D E (1981) *The Art of Computer Programming (Volume 2)* (2nd Edition) Addison–Wesley

Wilkinson J H (1965) *The Algebraic Eigenvalue Problem* Oxford University Press, Oxford

5 Parameters

1: **order** – Nag_OrderType *Input*

On entry: the **order** parameter 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 parameter.

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

2: **mode** – Integer *Input*

On entry: selects the operation to be performed:

mode $= 0$

Initialise **and** generate random numbers.

mode $= 1$

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

mode $= 2$

Generate random numbers using previously set up reference vector.

Constraint: $0 \leq$ **mode** ≤ 2.

3: **n** – Integer *Input*

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

Constraint: **n** > 0.

4: **xmu**[**n**] – const double *Input*

On entry: the vector of means, a, of the distribution.

5: **c**[dim] – const double *Input*

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

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

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

6: **ic** – Integer *Input*

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

Constraint: **ic** $\geq$ **n**.

7: **x**[**n**] – double *Output*

On exit: the pseudo-random multivariate Normal vector generated by the function.

8: **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).

9: **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.

10: **r**[dim] – double *Input/Output*

Note: the dimension, dim, of the array **r** must be at least $($**n**$+1)($**n**$+2)/2$.

On entry: if **mode** $= 2$, the reference vector as set up by nag_rngs_multi_normal (g05lzc) 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_multi_normal (g05lzc) with **mode** $= 2$.

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

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

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

On entry, **mode** = ⟨*value*⟩.
Constraint: $0 \leq$ **mode** ≤ 2.

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 1.

NE_INT_2

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

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

NE_POS_DEF

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

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ 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

The maximum absolute error in LL^T, and hence in the covariance matrix of the resulting vectors, is less than $(n \times \epsilon + (n + 3)\epsilon/2)$ times the maximum element of C, where ϵ is the ***machine precision***. Under normal circumstances, the above will be small compared to sampling error.

8 Further Comments

The time taken by nag_rngs_multi_normal (g05lzc) is of order n^3.

It is recommended that the diagonal elements of C should not differ too widely in order of magnitude. This may be achieved by scaling the variables if necessary. The actual matrix decomposed is $C + E = LL^T$, where E is a diagonal matrix with small positive diagonal elements. This ensures that, even when C is singular, or nearly singular, the Cholesky Factor L corresponds to a positive-definite covariance matrix that agrees with C within ***machine precision***.

9 Example

The example program prints two pseudo-random observations from a bivariate Normal distribution with means vector

$$\begin{bmatrix} 1.0 \\ 2.0 \end{bmatrix}$$

and covariance matrix

$$\begin{bmatrix} 2.0 & 1.0 \\ 1.0 & 3.0 \end{bmatrix},$$

generated by nag_rngs_multi_normal (g05lzc). The first observation is generated by a single call to nag_rngs_multi_normal (g05lzc) with **mode** $= 0$, and the second observation is generated using the same reference vector a call to nag_rngs_multi_normal (g05lzc) with **mode** $= 2$. The random number generator is initialised by nag_rngs_init_repeatable (g05kbc).

9.1 Program Text

```
/* nag_rngs_multi_normal(g05lzc) 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, nr;
  Integer  exit_status=0;
  Integer  pdc;
  NagError fail;
  Nag_OrderType order;

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

#ifdef NAG_COLUMN_MAJOR
#define C(I,J) c[(J-1)*pdc + I - 1]
  order = Nag_ColMajor;
#else
#define C(I,J) c[(I-1)*pdc + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);
  Vprintf("g05lzc Example Program Results\n\n");

  m = 2;
  nr = (m+1)*(m+2)/2;
  /* Allocate memory */
  if ( !(c = NAG_ALLOC(m * m, double)) ||
       !(r = NAG_ALLOC((m+1)*(m+2)/2, double)) ||
       !(x = NAG_ALLOC(m, double)) ||
       !(xmu = NAG_ALLOC(m, double)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
#ifdef NAG_COLUMN_MAJOR
  pdc = m;
#else
  pdc = m;
#endif

  /* 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);

  C(1, 1) = 2.0;
  C(2, 1) = 1.0;
  C(1, 2) = 1.0;
  C(2, 2) = 3.0;

  xmu[0] = 1.0;
  xmu[1] = 2.0;

  /* Set up reference vector and generate numbers */
  g05lzc(order, 0, m, xmu, c, m, x, igen, iseed, r, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from g05lzc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  for (j = 0; j < m; ++j)
    Vprintf("%10.4f\n",   x[j]);

  /* Generate numbers */
  g05lzc(order, 2, m, xmu, c, m, x, igen, iseed, r, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from g05lzc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  for (j = 0; j < m; ++j)
    Vprintf("%10.4f\n", x[j]);
 END:
  if (c) NAG_FREE(c);
  if (r) NAG_FREE(r);
  if (x) NAG_FREE(x);
  if (xmu) NAG_FREE(xmu);
  return exit_status;
}
```

9.2 Program Data

None.

9.3 Program Results

```
g05lzc Example Program Results

    3.9620
    2.4272
    0.9189
   -0.1605
```