NAG C Library Function Document

nag_rgsn_matrix_multi_normal (g05lyc)

1 Purpose

nag_rgsn_matrix_multi_normal (g05lyc) sets up a reference vector and generates an array of pseudo-random numbers from a multivariate Normal distribution with mean vector a and covariance matrix C.

2 Specification

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

3 Description

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

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

where m 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 $LL^{\mathrm{T}} = C$. L is not unique, if C is singular.

nag_rgsn_matrix_multi_normal (g05lyc) decomposes C to find such an L. It then stores m, 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 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_rgsn_matrix_multi_normal (g05lyc).

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 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: $0 \leq \mathbf{mode} \leq 2$.

3: **m** – Integer *Input*

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

Constraint: $\mathbf{m} > 0$.

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

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

5: **c**[dim] – const double *Input*

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

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

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

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***

6: **pdc** – Integer *Input*

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

Constraint: $\mathbf{pdc} \geq \mathbf{m}$.

7: **n** – Integer *Input*

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

Constraint: $\mathbf{n} \geq 1$.

8: **x**[dim] – double *Output*

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

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

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

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

On exit: the array of pseudo-random multivariate Normal vectors generated by the function, with $X(i,j)$ holding the jth dimension for the ith variate.

9: **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**.

10: **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).

11: **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.

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

On entry: if **mode** = 2, the reference vector as set up by nag_rgsn_matrix_multi_normal (g05lyc) 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_rgsn_matrix_multi_normal (g05lyc) with **mode** = 2.

13: **lr** – Integer *Input*

On entry: the dimension of the array **r** as declared in the function from which nag_rgsn_matrix_multi_normal (g05lyc) is called. If **mode** = 2, it must be the same as the value of **lr** specified in the prior call to nag_rgsn_matrix_multi_normal (g05lyc) with **mode** = 0 or 1.

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

14: **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** $= \langle value \rangle$.
Constraint: **pdx** > 0.

NE_INT_2

m is not the same as when **r** was set up in a previous call. Previous value $= \langle value \rangle$, **m** $= \langle value \rangle$.

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

On entry, **pdc** $= \langle value \rangle$, **m** $= \langle value \rangle$.
Constraint: **pdc** $\geq$ **m**.

On entry, **pdx** $= \langle value \rangle$, **m** $= \langle value \rangle$.
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

The maximum absolute error in LL^{T}, and hence in the covariance matrix of the resulting vectors, is less than $(m\epsilon + (m+3)\epsilon/2)$ 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_rgsn_matrix_multi_normal (g05lyc) is of order nm^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 $C + E = LL^{\mathrm{T}}$, 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 ten pseudo-random observations from a multivariate Normal distribution with means vector

$$\begin{bmatrix} 1.0 \\ 2.0 \\ -3.0 \\ 0.0 \end{bmatrix}$$

and 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_rgsn_matrix_multi_normal (g05lyc). All ten observations are generated by a single call to nag_rgsn_matrix_multi_normal (g05lyc) with **mode** $= 0$. The random number generator is initialized by nag_rngs_init_repeatable (g05kbc).

9.1 Program Text

```
/* nag_rngs_matrix_multi_normal (g05lyc) 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, *xmu=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

#define XMU(I) xmu[I - 1]

  INIT_FAIL(fail);
  exit_status = 0;

  /* Set the number of variables and variates */
  m = 4;
  n = 10;
  lr = m * (m + 1) + 2;
  pdc = m;
  pdx = m;

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

  Vprintf("nag_rgsn_matrix_multi_normal (g05lyc) 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;

  /* Input the means */
  XMU(1) = 1.0;
  XMU(2) = 2.0;
  XMU(3) = -3.0;
  XMU(4) = 0.0;

  /* Set the mode */
  mode = 0;

  /* Set up reference vector and generate n numbers */
  /* nag_rgsn_matrix_multi_normal (g05lyc).
   * Generates a matrix of random numbers from a multivariate
   * Normal distribution, seeds and generator passed
   * explicitly
   */
  nag_rgsn_matrix_multi_normal(order, mode, m, xmu, 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);
  if (xmu) NAG_FREE(xmu);

  return exit_status;
}
```

9.2 Program Data

None.

9.3 Program Results

```
nag_rgsn_matrix_multi_normal (g05lyc) Example Program Results

    3.7228    -7.4911    -7.9865     0.1579
    0.1335     7.2616    -2.7775    -0.1209
    0.9254    12.2770     0.8045    -0.0237
   -0.7430     1.2303    -1.0565    -0.0646
    0.6148   -15.3318     1.3101     0.0812
```

```
    1.1606    20.4061    -3.6213    -0.0935
   -0.0647    -2.0038     2.9313     0.0035
    1.8238     6.4318   -10.0483    -0.0348
    2.6441   -12.3753    -2.4718     0.1373
   -0.0049   -19.3144     1.1728     0.1233
```