NAG C Library Function Document

nag_rngs_corr_matrix (g05qbc)

1 Purpose

nag_rngs_corr_matrix (g05qbc) generates a random correlation matrix with given eigenvalues.

2 Specification

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

```
void nag_rngs_corr_matrix (Nag_OrderType order, Integer n, const double d[],
     double c[], Integer pdc, double eps, Integer igen, Integer iseed[],
     NagError *fail)
```

3 Description

Given n eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_n$, such that

$$\sum_{i=1}^{n} \lambda_i = n$$

and

$$\lambda_i \geq 0, \quad i = 1, 2, \ldots, n,$$

nag_rngs_corr_matrix (g05qbc) will generate a random correlation matrix, C, of dimension n, with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$.

The method used is based on that described by Lin and Bendel (1985). Let D be the diagonal matrix with values $\lambda_1, \lambda_2, \ldots, \lambda_n$ and let A be a random orthogonal matrix generated by nag_rngs_orthog_matrix (g05qac) then the matrix $C_0 = ADA^{\mathrm{T}}$ is a random covariance matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. The matrix C_0 is transformed into a correlation matrix by means of $n - 1$ elementary rotation matrices P_i such that $C = P_{n-1}P_{n-2} \ldots P_1 C_0 P_1^{\mathrm{T}} \ldots P_{n-2}^{\mathrm{T}} P_{n-1}^{\mathrm{T}}$. The restriction on the sum of eigenvalues implies that for any diagonal element of $C_0 > 1$, there is another diagonal element < 1. The P_i are constructed from such pairs, chosen at random, to produce a unit diagonal element corresponding to the first element. This is repeated until all diagonal elements are 1 to within a given tolerance ϵ.

The randomness of C should be interpreted only to the extent that A is a random orthogonal matrix and C is computed from A using the P_i which are chosen as arbitrarily as possible.

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_corr_matrix (g05qbc).

4 References

Lin S P and Bendel R B (1985) Algorithm AS213: Generation of population correlation on matrices with specified eigenvalues *Appl. Statist.* **34** 193–198

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: **n** – Integer *Input*

On entry: n, the dimension of the correlation matrix to be generated.

Constraint: **n** ≥ 1.

3: **d**[**n**] – const double *Input*

On entry: the n eigenvalues, λ_i, for $i = 1, 2, \ldots, n$.

Constraints:

$\mathbf{d}[i] \geq 0.0$, for $i = 0, 1, \ldots, n-1$;

$\sum_{i=1}^{n} \mathbf{d}[i] = n$ to within **eps**.

4: **c**[dim] – double *Output*

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

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 exit: a random correlation matrix, C, of dimension n.

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** $\geq$ **n**.

6: **eps** – double *Input*

On entry: ϵ, the maximum acceptable error in the diagonal elements.

Constraint: **eps** $\geq$ **n** $\times$ ***machineprecision*** (see Chapter x02).

Suggested value: **eps** = 0.00001.

7: **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).

8: **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.

9: **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 $\langle \mathit{value} \rangle$ had an illegal value.

NE_DIAG_ELEMENTS

The error in a diagonal element is greater than **eps**. The value of **eps** should be increased. Otherwise the program could be rerun with a different value used for the seed of the random number generator, see nag_rngs_init_repeatable (g05kbc) or nag_rngs_init_nonrepeatable (g05kcc).

NE_EIGVAL_SUM

On entry, the eigenvalues do not sum to **n**.

NE_INT

On entry, $\mathbf{n} = \langle \mathit{value} \rangle$.
Constraint: $\mathbf{n} \geq 1$.

On entry, $\mathbf{pdc} = \langle \mathit{value} \rangle$.
Constraint: $\mathbf{pdc} > 0$.

NE_INT_2

On entry, $\mathbf{pdc} = \langle \mathit{value} \rangle$, $\mathbf{n} = \langle \mathit{value} \rangle$.
Constraint: $\mathbf{pdc} \geq \mathbf{n}$.

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_NEGATIVE_EIGVAL

On entry, an eigenvalue is negative.

NE_REAL

On entry, $\mathbf{eps} < \mathbf{n} \times \boldsymbol{machineprecision}$: $\mathbf{eps} = \langle \mathit{value} \rangle$.

7 Accuracy

The maximum error in a diagonal element is given by **eps**.

8 Further Comments

The time taken by nag_rngs_corr_matrix (g05qbc) is approximately proportional to n^2.

9 Example

Following initialization of the pseudo-random number generator by a call to nag_rngs_init_repeatable (g05kbc), a 3 by 3 correlation matrix with eigenvalues of 0.7, 0.9 and 1.4 is generated and printed.

9.1 Program Text

```
/* nag_rngs_corr_matrix (g05qbc) 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 */
  double   eps;
  Integer  i, igen, j, n;
  Integer  exit_status=0;
  Integer  pdc;
  NagError fail;
  Nag_OrderType order;

  /* Arrays */
  double   *c=0, *d=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("nag_rngs_corr_matrix (g05qbc) Example Program Results\n\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%ld%*[^\n] ", &n);

  /* Allocate memory */
  if ( !(c = NAG_ALLOC(10 * 10, double)) ||
       !(d = NAG_ALLOC(10, double)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

#ifdef NAG_COLUMN_MAJOR
  pdc = n;
#else
  pdc = n;
#endif

  if (n <= 10)
    {

      for (i = 0; i < n; ++i)
        {
          Vscanf("%lf", &d[i]);
        }
      Vscanf("%*[^\n] ");

      eps = 1e-4;

      /* igen identifies the stream. */
      igen = 1;
      /* Initialise the seed to a repeatable sequence */
      iseed[0] = 1762543;
      iseed[1] = 9324783;
      iseed[2] = 423446;
      iseed[3] = 742355;

      /* 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);

      /* nag_rngs_corr_matrix (g05qbc).
       * Computes a random correlation matrix
       */
      nag_rngs_corr_matrix(order, n, d, c, pdc, eps, igen, iseed, &fail);
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from nag_rngs_corr_matrix (g05qbc).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= n; ++j)
            {
              Vprintf("%9.3f%s", C(i,j), j%3 == 0 || j == n ?"\n":" ");
            }
        }
    }
 END:
  if (c) NAG_FREE(c);
  if (d) NAG_FREE(d);
  return exit_status;
}
```

9.2 Program Data

```
nag_rngs_corr_matrix (g05qbc) Example Program Data
3
0.7 0.9 1.4
```

9.3 Program Results

```
nag_rngs_corr_matrix (g05qbc) Example Program Results

    1.000     0.204    -0.106
    0.204     1.000    -0.278
   -0.106    -0.278     1.000
```