NAG Library Function Document

nag_rand_corr_matrix (g05pyc)

1 Purpose

nag_rand_corr_matrix (g05pyc) generates a random correlation matrix with given eigenvalues.

2 Specification

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

```
void nag_rand_corr_matrix (Integer n, const double d[], double eps,
     Integer state[], double c[], Integer pdc, 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_rand_corr_matrix (g05pyc) 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_rand_orthog_matrix (g05pxc) 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_rand_init_repeatable (g05kfc) (for a repeatable sequence if computed sequentially) or nag_rand_init_nonrepeatable (g05kgc) (for a non-repeatable sequence) must be called prior to the first call to nag_rand_corr_matrix (g05pyc).

4 References

Lin S P and Bendel R B (1985) Algorithm AS 213: Generation of population correlation on matrices with specified eigenvalues *Appl. Statist.* **34** 193–198

5 Arguments

1: **n** – Integer *Input*

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

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

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

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

Constraints:

$\mathbf{d}[i-1] \geq 0.0$, for $i = 1, 2, \ldots, n$;
$\sum_{i=1}^{n} \mathbf{d}[i-1] = n$ to within **eps**.

3: **eps** – double *Input*

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

Suggested value: **eps** $= 0.00001$.

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

4: **state**[*dim*] – Integer *Communication Array*

Note: the dimension, dim, of this array is dictated by the requirements of associated functions that must have been previously called. This array MUST be the same array passed as argument **state** in the previous call to nag_rand_init_repeatable (g05kfc) or nag_rand_init_nonrepeatable (g05kgc).

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

5: **c**[**n** $\times$ **pdc**] – double *Output*

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

6: **pdc** – Integer *Input*

On entry: the stride separating row elements of the matrix C in the array **c**.

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

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

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

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

NE_DIAG_ELEMENTS

The diagonals of the returned matrix are not unity, try increasing the value of **eps**, or rerun the code using a different seed.

NE_EIGVAL_SUM

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

NE_INT

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 1.

On entry, **pdc** $= \langle value \rangle$.
Constraint: **pdc** > 0.

NE_INT_2

On entry, **pdc** $= \langle value \rangle$ and **n** $= \langle value \rangle$.
Constraint: **pdc** $\geq$ **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 contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 3.6.6 in How to Use the NAG Library and its Documentation for further information.

NE_INVALID_STATE

On entry, **state** vector has been corrupted or not initialized.

NE_NEGATIVE_EIGVAL

On entry, an eigenvalue is negative.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in How to Use the NAG Library and its Documentation for further information.

NE_REAL

On entry, **eps** $= \langle value \rangle$.
Constraint: **eps** $\geq$ **n** $\times$ ***machine precision***.

7 Accuracy

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

8 Parallelism and Performance

nag_rand_corr_matrix (g05pyc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_rand_corr_matrix (g05pyc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the x06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

The time taken by nag_rand_corr_matrix (g05pyc) is approximately proportional to n^2.

10 Example

Following initialization of the pseudorandom number generator by a call to nag_rand_init_repeatable (g05kfc), a 3 by 3 correlation matrix with eigenvalues of 0.7, 0.9 and 1.4 is generated and printed.

10.1 Program Text

```
/* nag_rand_corr_matrix (g05pyc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>

#define C(I, J) c[J*pdc + I]

int main(void)
{
  /* Integer scalar and array declarations */
  Integer exit_status = 0;
  Integer i, j, lstate, n, c_size;
  Integer *state = 0;
  Integer pdc;
  /* NAG structures */
  NagError fail;
  /* Double scalar and array declarations */
  double *c = 0, *d = 0;
  /* Set tolerance */
  double eps = 0.00001e0;
  /* Choose the base generator */
  Nag_BaseRNG genid = Nag_Basic;
  Integer subid = 0;
  /* Set the seed */
  Integer seed[] = { 1762543 };
  Integer lseed = 1;

  /* Initialize the error structure */
  INIT_FAIL(fail);

  printf("nag_rand_corr_matrix (g05pyc) Example Program Results\n\n");

  /* Get the length of the state array */
  lstate = -1;
  nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Read data from a file */
  /* Skip heading */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  /* Read in initial parameters */
#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%*[^\n] ", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n] ", &n);
#endif

  pdc = n;
  c_size = pdc * n;

  /* Allocate arrays */
  if (!(c = NAG_ALLOC(c_size, double)) ||
      !(d = NAG_ALLOC(n, double)) || !(state = NAG_ALLOC(lstate, Integer)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

  /* Read in the eigenvalues */
  for (i = 0; i < n; i++)
#ifdef _WIN32
    scanf_s("%lf ", &d[i]);
#else
    scanf("%lf ", &d[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* Initialize the generator to a repeatable sequence */
  nag_rand_init_repeatable(genid, subid, seed, lseed, state, &lstate, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_init_repeatable (g05kfc).\n%s\n",
           fail.message);
    exit_status = 1;
    goto END;
  }

  /* Generate the correlation matrix */
  nag_rand_corr_matrix(n, d, eps, state, c, pdc, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_rand_corr_matrix (g05pyc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Display the results */
  for (i = 0; i < n; i++) {
    printf("  ");
    for (j = 0; j < n; j++)
      printf("%8.3f%s", C(i, j), (j + 1) % 10 ? " " : "\n");
    if (n % 10)
      printf("\n");
  }

END:
  NAG_FREE(c);
  NAG_FREE(d);
  NAG_FREE(state);

  return exit_status;
}
```

10.2 Program Data

```
nag_rand_corr_matrix (g05pyc) Example Program Data
3
0.7 0.9 1.4
```

10.3 Program Results

```
nag_rand_corr_matrix (g05pyc) Example Program Results

     1.000   -0.255   -0.100
    -0.255    1.000    0.234
    -0.100    0.234    1.000
```