NAG Library Function Document

nag_rand_bivariate_copula_clayton (g05rec)

1 Purpose

nag_rand_bivariate_copula_clayton (g05rec) generates pseudorandom uniform bivariates with joint distribution of a Clayton/Cook–Johnson Archimedean copula.

2 Specification

```
#include <nag.h>
#include <nagg05.h>
void nag_rand_bivariate_copula_clayton (Nag_OrderType order,
     Integer state[], double theta, Integer n, double x[], Integer pdx,
     Integer sdx, NagError *fail)
```

3 Description

Generates pseudorandom uniform bivariates $\{u_1, u_2\} \in (0,1]^2$ whose joint distribution is the Clayton/ Cook–Johnson Archimedean copula C_θ with parameter θ, given by

$$C_\theta = \left[\max\left({u_1}^{-\theta} + {u_2}^{-\theta} - 1, 0\right)\right]^{-1/\theta}, \quad \theta \in (-1, \infty) \setminus \{0\}$$

with the special cases:

$C_{-1} = \max(u_1 + u_2 - 1, 0)$, the Fréchet–Hoeffding lower bound;

$C_0 = u_1 u_2$, the product copula;

$C_\infty = \min(u_1, u_2)$, the Fréchet–Hoeffding upper bound.

The generation method uses conditional sampling.

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_bivariate_copula_clayton (g05rec).

4 References

Nelsen R B (2006) *An Introduction to Copulas* (2nd Edition) Springer Series in Statistics

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 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

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

2: **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.

3: **theta** – double *Input*

On entry: θ, the copula parameter.

Constraint: **theta** ≥ -1.0.

4: **n** – Integer *Input*

On entry: n, the number of bivariates to generate.

Constraint: **n** ≥ 0.

5: **x**[**pdx** $\times$ **sdx**] – double *Output*

Note: where $\mathbf{X}(i,j)$ appears in this document, it refers to the array element $\mathbf{x}[(j-1) \times \mathbf{pdx} + i - 1]$.

On exit: the n bivariate uniforms with joint distribution described by C_θ, with $\mathbf{X}(i,j)$ holding the ith value for the jth dimension if **order** = Nag_ColMajor and the jth value for the ith dimension of **order** = Nag_RowMajor.

6: **pdx** – Integer *Input*

On entry: the stride separating matrix row elements in the array **x**.

Constraints:

if **order** = Nag_ColMajor, **pdx** $\geq$ **n**;
if **order** = Nag_RowMajor, **pdx** ≥ 2.

7: **sdx** – Integer *Input*

On entry: the secondary dimension of **X**.

Constraints:

if **order** = Nag_ColMajor, **sdx** ≥ 2;
if **order** = Nag_RowMajor, **sdx** $\geq$ **n**.

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

The NAG error argument (see Section 3.6 in 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, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

NE_INT_2

On entry, **pdx** must be at least ⟨*value*⟩: **pdx** = ⟨*value*⟩.

On entry, **sdx** must be at least ⟨*value*⟩: **sdx** = ⟨*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 contact NAG for assistance.

NE_INVALID_STATE

On entry, corrupt **state** argument.

NE_REAL

On entry, invalid **theta**: **theta** $= \langle value \rangle$.
Constraint: **theta** ≥ -1.0.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_rand_bivariate_copula_clayton (g05rec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

In practice, the need for numerical stability restricts the range of θ such that:

if $(\theta + 1) < \epsilon$, the function returns pseudorandom uniform variates with C_{-1} joint distribution;

if $|\theta| < 1.0 \times 10^{-6}$, the function returns pseudorandom uniform variates with C_0 joint distribution;

if $\theta > \ln \epsilon_s / \ln\left(1.0 \times 10^{-2}\right)$, the function returns pseudorandom uniform variates with C_∞ joint distribution;

where ϵ_s is the safe-range parameter, the value of which is returned by nag_real_safe_small_number (X02AMC); and ϵ is the ***machine precision*** returned by nag_machine_precision (X02AJC).

10 Example

This example generates thirteen variates for copula $C_{-0.8}$.

10.1 Program Text

```
/* nag_rand_bivariate_copula_clayton (g05rec) Example Program.
 *
 * Copyright 2009, Numerical Algorithms Group.
 *
 * Mark 9, 2009.
 */
/* Pre-processor includes */
#include <stdio.h>
#include <math.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg05.h>


#define X(I, J) x[order == Nag_ColMajor?((J-1)*pdx + I-1):((I-1)*pdx + J-1)]

int main(void)
{
  /* Integer scalar and array declarations */
```

```
  Integer       exit_status = 0;
  Integer       i, lstate, pdx, sdx;
  Integer       *state = 0;

  /* Double scalar and array declarations */
  double        *x = 0;

  /* NAG structures */
  NagError      fail;

  /* Use row major order */
  Nag_OrderType order = Nag_RowMajor;

  /* Set the number of variates */
  Integer       n = 13;

  /* Choose the base generator */
  Nag_BaseRNG   genid = Nag_Basic;
  Integer       subid = 0;

  /* Set the seed */
  Integer       seed[] = { 1762543 };
  Integer       lseed = 1;

  /* Set the theta parameter value */
  double        theta = -0.8e0;

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

  printf(
          "nag_rand_bivariate_copula_clayton (g05rec) "
          "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;
    }

  /* Set matrix size and principal dimension according to storage order */
  pdx = (order == Nag_ColMajor)?n:2;
  sdx = (order == Nag_ColMajor)?2:n;

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

  /* Initialise 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 variates */
  nag_rand_bivariate_copula_clayton(order, state, theta, n, x, pdx, sdx,
                                    &fail);
```

```
  if (fail.code != NE_NOERROR)
    {
      printf("Error from "
              "nag_rand_bivariate_copula_clayton (g05rec).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Display the results */
  printf("Uniform variates with copula joint distrbution\n");
  for (i = 1; i <= n; i++)
    {
      printf(" %9.6f   %9.6f\n", X(i, 1), X(i, 2));
    }

 END:
  NAG_FREE(x);
  NAG_FREE(state);

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_rand_bivariate_copula_clayton (g05rec) Example Program Results

 Uniform variates with copula joint distrbution
 0.640009    0.222257
 0.115415    0.810119
 0.748575    0.143920
 0.800287    0.106173
 0.113547    0.994596
 0.497526    0.765548
 0.390418    0.492506
 0.789199    0.119611
 0.503205    0.411606
 0.674986    0.209262
 0.060032    0.905477
 0.265450    0.708476
 0.627568    0.237012
```