# NAG Library Function Document

# nag_rand_copula_clayton (g05rhc)

## 1 Purpose

nag_rand_copula_clayton (g05rhc) generates pseudorandom uniform variates with joint distribution of a Clayton/Cook–Johnson Archimedean copula.

## 2 Specification

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

## 3 Description

Generates $n$ pseudorandom uniform $m$-variates whose joint distribution is the Clayton/Cook–Johnson Archimedean copula $C_\theta$, given by

$$C_\theta = \left(u_1^{-\theta} + u_2^{-\theta} + \cdots + u_m^{-\theta} - m + 1\right)^{-1/\theta}, \quad \begin{cases} \theta \in (0, \infty), \\ u_j \in (0, 1], \quad j = 1, \ldots m; \end{cases}$$

with the special case:

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

The generation method uses mixture of powers.

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_copula_clayton (g05rhc).

## 4 References

Marshall A W and Olkin I (1988) Families of multivariate distributions *Journal of the American Statistical Association* **83** 403

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*: $\theta$, the copula parameter.

*Constraint*: **theta** $\geq 1.0 \times 10^{-6}$.

4: **n** – Integer *Input*

*On entry*: $n$, the number of pseudorandom uniform variates to generate.

*Constraint*: **n** $\geq 0$.

5: **m** – Integer *Input*

*On entry*: $m$, the number of dimensions.

*Constraint*: **m** $\geq 2$.

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

**Note**: where $\mathbf{X}(i,j)$ appears in this document, it refers to the array element

**x**$[(j-1) \times$ **pdx** $+ i - 1]$ when **order** = Nag_ColMajor;
**x**$[(i-1) \times$ **pdx** $+ j - 1]$ when **order** = Nag_RowMajor.

*On exit*: the pseudorandom uniform variates with joint distribution described by $C_\theta$, with $\mathbf{X}(i,j)$ holding the $i$th value for the $j$th dimension if **order** = Nag_ColMajor and the $j$th value for the $i$th dimension of **order** = Nag_RowMajor.

7: **pdx** – Integer *Input*

*On entry*: the stride separating 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**.

8: **sdx** – Integer *Input*

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

*Constraints*:

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

9: **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.
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE_BAD_PARAM**

On entry, argument ⟨*value*⟩ had an illegal value.

**NE_INT**

On entry, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{m} \geq 2$.

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

**NE_INT_2**

On entry, **pdx** must be at least $\langle value \rangle$: $\mathbf{pdx} = \langle value \rangle$.

On entry, **sdx** must be at least $\langle value \rangle$: $\mathbf{sdx} = \langle value \rangle$.

**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 the Essential Introduction for further information.

**NE_INVALID_STATE**

On entry, corrupt **state** argument.

**NE_NO_LICENCE**

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

**NE_REAL**

On entry, invalid **theta**: $\mathbf{theta} = \langle value \rangle$.
Constraint: $\mathbf{theta} \geq 1.0 \times 10^{-6}$.

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

nag_rand_copula_clayton (g05rhc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

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' Note for your implementation for any additional implementation-specific information.

## 9 Further Comments

In practice, the need for numerical stability restricts the range of $\theta$ such that:

the function requires $\theta \geq 1.0 \times 10^{-6}$;

if $\theta > 200.0$, the function returns pseudorandom uniform variates with $C_{\infty}$ joint distribution.

## 10 Example

This example generates thirteen four-dimensional variates for copula $C_{1.3}$.

## 10.1 Program Text

```
/* nag_rand_copula_clayton (g05rhc) Example Program.
 *
 * Copyright 2014 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, j, 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 variables and variates */
  Integer       n = 13, m = 4;

  /* 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 = 1.3e0;

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

  printf(
          "nag_rand_copula_clayton (g05rhc) 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:m;
  sdx = (order == Nag_ColMajor)?m: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_copula_clayton(order, state, theta, n, m, x, pdx, sdx, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rand_copula_clayton (g05rhc).\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("  ");
      for (j = 1; j <= m; j++)
        printf("%9.6f%s", X(i, j), j < m?" ":"\n");
    }

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

  return exit_status;
}
```

## 10.2 Program Data

None.

## 10.3 Program Results

```
nag_rand_copula_clayton (g05rhc) Example Program Results

 Uniform variates with copula joint distrbution
 0.857596   0.504767   0.976144   0.589532
 0.318626   0.637221   0.995855   0.589750
 0.904978   0.695043   0.935327   0.932918
 0.527790   0.180439   0.417668   0.232986
 0.150999   0.977657   0.262118   0.386696
 0.390568   0.793755   0.307329   0.314952
 0.127882   0.170887   0.175130   0.056834
 0.761251   0.431446   0.349758   0.291269
 0.387054   0.442952   0.360993   0.377364
 0.124195   0.064710   0.047215   0.078048
 0.686601   0.950027   0.928913   0.976284
 0.525855   0.821820   0.713407   0.491447
 0.095474   0.045934   0.126494   0.194655
```