NAG Library Function Document

nag_rand_dirichlet (g05sec)

1 Purpose

nag_rand_dirichlet (g05sec) generates a vector of pseudorandom numbers taken from a Dirichlet distribution.

2 Specification

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

3 Description

The distribution has PDF (probability density function)

$$f(x) = \frac{1}{B(\alpha)}\prod_{i=1}^{m} x_i^{\alpha_i - 1} \quad \text{and}$$

$$B(\alpha) = \frac{\prod_{i=1}^{m}\Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{m}\alpha_i\right)}$$

where $x = \{x_1, x_2, \ldots, x_m\}$ is a vector of dimension m, such that $x_i > 0$ for all i and $\sum_{i=1}^{m} x_i = 1$.

nag_rand_dirichlet (g05sec) generates a draw from a Dirichlet distribution by first drawing m independent samples, $y_i \sim \text{gamma}(\alpha_i, 1)$, i.e., independent draws from a gamma distribution with parameters $\alpha_i > 0$ and one, and then setting $x_i = y_i / \sum_{j=1}^{m} y_j$.

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_dirichlet (g05sec).

4 References

Dagpunar J (1988) *Principles of Random Variate Generation* Oxford University Press

Hastings N A J and Peacock J B (1975) *Statistical Distributions* Butterworth

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: **n** – Integer *Input*

On entry: n, the number of pseudorandom numbers to be generated.

Constraint: $\mathbf{n} \geq 0$.

3: **m** – Integer *Input*

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

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

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

On entry: the parameter vector for the distribution.

Constraint: $\mathbf{a}[i-1] > 0.0$, for $i = 1, 2, \ldots, \mathbf{m}$.

5: **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.

6: **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.

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

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

On exit: the n pseudorandom numbers from the specified Dirichlet distribution, with $\mathbf{X}(i, j)$ holding the jth dimension for the ith variate.

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, $\mathbf{pdx} \geq \mathbf{n}$;
if **order** = Nag_RowMajor, $\mathbf{pdx} \geq \mathbf{m}$.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

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

NE_INT

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

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

NE_INT_2

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

On entry, $\mathbf{pdx} = \langle value \rangle$ and $\mathbf{n} = \langle value \rangle$.
Constraint: **order** = Nag_ColMajor or $\mathbf{pdx} \geq \mathbf{n}$.

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 contact NAG for assistance.

NE_INVALID_STATE

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

NE_REAL_ARRAY

On entry, at least one $\mathbf{a}[i] \leq 0$.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_rand_dirichlet (g05sec) 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

None.

10 Example

This example prints a set of five pseudorandom numbers from a Dirichlet distribution with parameters $m = 4$ and $\alpha = \{2.0, 2.0, 2.0, 2.0\}$, generated by a single call to nag_rand_dirichlet (g05sec), after initialization by nag_rand_init_repeatable (g05kfc).

10.1 Program Text

```
/* nag_rand_dirichlet (g05sec) Example Program.
 *
 * Copyright 2008, 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*pdx + I):(I*pdx + J)]
```

```
int main(void)
{
  /* Integer scalar and array declarations */
  Integer       exit_status = 0;
  Integer       pdx, x_size, i, j, lstate;
  Integer       *state = 0;

  /* NAG structures */
  NagError      fail;

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

  /* Set the distribution parameters */
  double        a[] = { 2.0e0, 2.0e0, 2.0e0, 2.0e0 };
  Integer       m = 4;

  /* Set the sample size */
  Integer       n = 5;


  /* Return the results in column major order */
  Nag_OrderType order = Nag_ColMajor;

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

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

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

  printf("nag_rand_dirichlet (g05sec) 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;
    }

  pdx = (order == Nag_ColMajor)?n:m;
  x_size = (order == Nag_ColMajor)?pdx * m:pdx * n;

  /* Allocate arrays */
  if (!(x = NAG_ALLOC(x_size, 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 the variates*/
```

```
  nag_rand_dirichlet(order, n, m, a, state, x, pdx, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_rand_dirichlet (g05sec).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  /* Display the variates*/
  for (i = 0; i < n; i++)
    {
      for (j = 0; j < m; j++)
        printf("%10.4f", X(i, j));
      printf("\n");
    }

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

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_rand_dirichlet (g05sec) Example Program Results

    0.3600    0.3138    0.0837    0.2426
    0.2874    0.5121    0.1497    0.0509
    0.2286    0.2190    0.3959    0.1566
    0.1744    0.3961    0.2764    0.1530
    0.1522    0.2845    0.2074    0.3559
```