NAG C Library Function Document

nag_rngs_exp_mix (g05lqc)

1 Purpose

nag_rngs_exp_mix (g05lqc) generates a vector of pseudo-random numbers from an exponential mix distribution composed of m exponential distributions each having a mean a_i and weight w_i.

2 Specification

```
void nag_rngs_exp_mix (Integer nmix, const double a[], const double wgt[],
     Integer n, double x[], Integer igen, Integer iseed[], NagError *fail)
```

3 Description

The distribution has PDF (probability density function)

$$f(x) = \begin{cases} \sum_{i=1}^{m} \frac{1}{a_i} w_i e^{-x/a_i} & \text{if } x > 0, \\ 0 & \text{otherwise,} \end{cases}$$

where $\sum_{i=1}^{m} w_i = 1$ and $a_i > 0$, $w_i \geq 0$.

nag_rngs_exp_mix (g05lqc) returns the values x_i by selecting, with probability w_j, random variates from an exponential distribution with parameter a_j.

One of the initialisation 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_exp_mix (g05lqc).

4 References

Knuth D E (1981) *The Art of Computer Programming (Volume 2)* (2nd Edition) Addison–Wesley

Kendall M G and Stuart A (1969) *The Advanced Theory of Statistics (Volume 1)* (3rd Edition) Griffin

5 Parameters

1: **nmix** – Integer *Input*

On entry: the number, m, of exponential distributions in the mix.

Constraint: **nmix** ≥ 1.

2: **a**[**nmix**] – const double *Input*

On entry: the m parameters a_i for the m exponential distributions in the mix.

Constraint: **a**$[i] > 0.0$ for $i = 0, 1, \ldots,$ **nmix** $- 1$.

3: **wgt**[**nmix**] – double *Input/Output*

On entry: the m weights w_i for the m exponential distributions in the mix.

On exit: used as internal workspace prior to being restored and hence is unchanged.

Constraints:

$\sum_{i=1}^{\mathbf{nmix}} \mathbf{wgt}[i] = 1.0$;
wgt$[i] \geq 0.0$ for $i = 0, 1, \ldots,$ **nmix** $- 1$.

4: **n** – Integer *Input*

On entry: the number, n, of pseudo-random numbers to be generated.

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

5: **x**[dim] – double *Output*

Note: the dimension, dim, of the array **x** must be at least $\max(1, \mathbf{n})$.

On exit: the n pseudo-random numbers from the specified exponential mix distribution.

6: **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 initialisation by a prior call to one of the functions nag_rngs_init_repeatable (g05kbc) or nag_rngs_init_nonrepeatable (g05kcc).

7: **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.

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

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

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

NE_REAL

On entry, sum of weights **wgt** is not equal to 1.0: sum $= \langle value \rangle$.

NE_REAL_ARRAY_ELEM_CONS

On entry, $\mathbf{wgt}[i-1] < 0.0$: $i = \langle value \rangle$, $\mathbf{wgt}[i-1] = \langle value \rangle$.

On entry, $\mathbf{a}[i-1] \leq 0.0$: $i = \langle value \rangle$, $\mathbf{a}[i-1] = \langle value \rangle$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter $\langle value \rangle$ had an illegal 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 consult NAG for assistance.

7 Accuracy

Not applicable.

8 Further Comments

None.

9 Example

The example program prints the first five pseudo-random real numbers from an exponential mix distribution comprising three exponential distributions with parameters $a_1 = 1.0$, $a_2 = 5.0$ and $a_3 = 2.0$, and with respective weights 0.5, 0.3 and 0.2. The numbers are generated by a single call to nag_rngs_exp_mix (g05lqc), after initialisation by nag_rngs_init_repeatable (g05kbc).

9.1 Program Text

```
/* nag_rngs_exp_mix(g05lqc) 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 */
  Integer  igen, j, m, nmix;
  Integer  exit_status=0;
  NagError fail;

  /* Arrays */
  double   *a=0, *wgt=0, *x=0;
  Integer  iseed[4];

  INIT_FAIL(fail);
  Vprintf("g05lqc Example Program Results\n\n");

  m = 5;
  nmix = 3;
  /* Allocate memory */
  if ( !(a = NAG_ALLOC(nmix, double)) ||
       !(wgt = NAG_ALLOC(nmix, double)) ||
       !(x = NAG_ALLOC(m, double)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Initialise the seed to a repeatable sequence */
  iseed[0] = 1762543;
  iseed[1] = 9324783;
  iseed[2] = 42344;
  iseed[3] = 742355;
  /* igen identifies the stream. */
  igen = 1;
  g05kbc(&igen, iseed);

  a[0] = 1.0;
  a[1] = 5.0;
  a[2] = 2.0;

  wgt[0] = 0.5;
  wgt[1] = 0.3;
  wgt[2] = 0.2;
```

```
  g05lqc(nmix, a, wgt, m, x, igen, iseed, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from g05lqc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  for (j = 0; j < m; ++j)
    {
      Vprintf("%10.4f\n",   x[j]);
    }
 END:
  if (a) NAG_FREE(a);
  if (wgt) NAG_FREE(wgt);
  if (x) NAG_FREE(x);
  return exit_status;
}
```

9.2 Program Data

None.

9.3 Program Results

```
g05lqc Example Program Results

    0.8275
    1.0723
    0.9284
    5.4923
    0.1827
```