NAG Library Function Document

nag_rand_exp (g05sfc)

1 Purpose
nag_rand_exp (g05sfc) generates a vector of pseudorandom numbers from a (negative) exponential distribution with mean \( a \).

2 Specification
```c
#include <nag.h>
#include <nagg05.h>
void nag_rand_exp (Integer n, double a, Integer state[], double x[],
                  NagError *fail)
```

3 Description
The exponential distribution has PDF (probability density function):
\[
  f(x) = \frac{1}{a}e^{-x/a} \quad \text{if } x \geq 0,
\]
\[
  f(x) = 0 \quad \text{otherwise}.
\]
nag_rand_exp (g05sfc) returns the values
\[
  x_i = -a \ln y_i
\]
where \( y_i \) are the next \( n \) numbers generated by a uniform \((0,1]\) generator.

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_exp (g05sfc).

4 References

5 Arguments
1: \( n \) – Integer
   *Input*
   On entry: \( n \), the number of pseudorandom numbers to be generated.
   Constraint: \( n \geq 0 \).
2: \( a \) – double
   *Input*
   On entry: \( a \), the mean of the distribution.
   Constraint: \( a > 0.0 \).
3: \( \text{state}[\text{dim}] \) – Integer
   *Communication Array*
   Note: the dimension, \( \text{dim} \), 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 \( \text{state} \) 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.

4: \( x[n] \) – double

Output

On exit: the \( n \) pseudorandom numbers from the specified exponential distribution.

5: \( \text{fail} \) – 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, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

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

**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, \( a = \langle \text{value} \rangle \).

Constraint: \( a > 0.0 \).

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_rand_exp (g05sfc) 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

None.

10 Example

This example prints five pseudorandom numbers from an exponential distribution with mean 1.0, generated by a single call to nag_rand_exp (g05sfc), after initialization by nag_rand_init_repeatable (g05kfc).

10.1 Program Text

```c
/* nag_rand_exp (g05sfc) 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 <nag05.h>

int main(void)
{
    /* Integer scalar and array declarations */
    Integer exit_status = 0;
    Integer i, lstate;
    Integer *state = 0;
    /* NAG structures */
    NagError fail;
    /* Double scalar and array declarations */
    double *x = 0;
    /* Set the distribution parameters */
    double a = 1.0e0;
    /* Set the sample size */
    Integer n = 5;
    /* 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_exp (g05sfc) 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;
    }
```

Mark 25
/* Allocate arrays */
if (!((x = NAG_ALLOC(n, 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_exp(n, a, state, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_exp (g05sfc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Display the variates*/
for (i = 0; i < n; i++)
    printf("%10.4f\n", x[i]);

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

10.2 Program Data
None.

10.3 Program Results
nag_rand_exp (g05sfc) Example Program Results

<table>
<thead>
<tr>
<th>Variate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4520</td>
<td></td>
</tr>
<tr>
<td>2.2398</td>
<td></td>
</tr>
<tr>
<td>0.2930</td>
<td></td>
</tr>
<tr>
<td>0.2253</td>
<td></td>
</tr>
<tr>
<td>2.2577</td>
<td></td>
</tr>
</tbody>
</table>