

NAG Library Function Document

nag_rand_triangular (g05spc)

1 Purpose

nag_rand_triangular (g05spc) generates a vector of pseudorandom numbers from a triangular distribution with parameters x_{\min} , x_{med} and x_{\max} .

2 Specification

```
#include <nag.h>
#include <nagg05.h>

void nag_rand_triangular (Integer n, double xmin, double xmed, double xmax,
    Integer state[], double x[], NagError *fail)
```

3 Description

The triangular distribution has a PDF (probability density function) that is triangular in profile. The base of the triangle ranges from $x = x_{\min}$ to $x = x_{\max}$ and the PDF has a maximum value of $\frac{2}{x_{\max} - x_{\min}}$ at $x = x_{\text{med}}$. If $x_{\min} = x_{\text{med}} = x_{\max}$ then $x = x_{\text{med}}$ with probability 1; otherwise the triangular distribution has PDF:

$$f(x) = \frac{x - x_{\min}}{x_{\text{med}} - x_{\min}} \times \frac{2}{x_{\max} - x_{\min}} \quad \text{if } x_{\min} \leq x \leq x_{\text{med}},$$

$$f(x) = \frac{x_{\max} - x}{x_{\max} - x_{\text{med}}} \times \frac{2}{x_{\max} - x_{\min}} \quad \text{if } x_{\text{med}} < x \leq x_{\max},$$

$$f(x) = 0 \quad \text{otherwise.}$$

One of the initialization functions nag_rand_init_repeatable (g05kfc) (for a repeatable sequence if computed sequentially) or nag_rand_init_nonrepeatable (g05kge) (for a non-repeatable sequence) must be called prior to the first call to nag_rand_triangular (g05spc).

4 References

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

5 Arguments

- 1: **n** – Integer *Input*
On entry: n , the number of pseudorandom numbers to be generated.
Constraint: $n \geq 0$.
- 2: **xmin** – double *Input*
On entry: the end point x_{\min} of the triangular distribution.

- 3: **xmed** – double *Input*
On entry: the median of the distribution x_{med} (also the location of the vertex of the triangular distribution at which the PDF reaches a maximum).
Constraint: **xmed** \geq **xmin**.
- 4: **xmax** – double *Input*
On entry: the end point x_{max} of the triangular distribution.
Constraint: **xmax** \geq **xmed**.
- 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**[**n**] – double *Output*
On exit: the *n* pseudorandom numbers from the specified triangular distribution.
- 7: **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 $\langle \text{value} \rangle$ 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_2

On entry, **xmax** = $\langle value \rangle$ and **xmed** = $\langle value \rangle$.
 Constraint: **xmax** \geq **xmed**.

On entry, **xmed** = $\langle value \rangle$ and **xmin** = $\langle value \rangle$.
 Constraint: **xmed** \geq **xmin**.

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag_rand_triangular (g05spc) 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 a triangular distribution with parameters $x_{\min} = -1.0$, $x_{\text{med}} = 0.5$ and $x_{\max} = 1.0$, generated by a single call to nag_rand_triangular (g05spc), after initialization by nag_rand_init_repeatab (g05kfc).

10.1 Program Text

```

/* nag_rand_triangular (g05spc) 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>

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       xmin = -1.0e0;
  double       xmax = 1.0e0;
  double       xmed = 0.50e0;

  /* 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_triangular (g05spc) 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;
}

/* 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_triangular(n, xmin, xmed, xmax, state, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_triangular (g05spc).\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_triangular (g05spc) Example Program Results

```
0.3817  
-0.4348  
0.4960  
0.5509  
-0.4398
```
