NAG Library Function Document

nag_rand_uniform (g05sqc)

1 Purpose

nag_rand_uniform (g05sqc) generates a vector of pseudorandom numbers uniformly distributed over the interval $[a, b]$.

2 Specification

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

3 Description

If $a = 0$ and $b = 1$, nag_rand_uniform (g05sqc) returns the next $n$ values $y_i$ from a uniform $(0, 1)$ generator (see nag_rand_basic (g05sac) for details).

For other values of $a$ and $b$, nag_rand_uniform (g05sqc) applies the transformation

$$x_i = a + (b - a)y_i.$$ 

The function ensures that the values $x_i$ lie in the closed interval $[a, b]$.

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_uniform (g05sqc).

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**

3:  
   **b** – double  
   **Input**

   On entry: the end points $a$ and $b$ of the uniform distribution.
   
   Constraint: $a \leq b$.

4:  
   **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.
5: \( x[n] \) – double
   \( On \ exit: \) the \( n \) pseudorandom numbers from the specified uniform distribution.

6: \( \text{fail} \) – NagError *
   \( Input/Output \)
   The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \( (\text{value}) \) had an illegal value.

\textbf{NE_INT}

On entry, \( n = (\text{value}) \).
Constraint: \( n \geq 0 \).

\textbf{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.

\textbf{NE_INVALID_STATE}

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

\textbf{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.

\textbf{NE_REAL_2}

On entry, \( a = (\text{value}) \) and \( b = (\text{value}) \).
Constraint: \( b \geq a \).

7 Accuracy

Not applicable.

8 Parallelism and Performance

nag\_rand\_uniform (g05sqc) 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

Although $y_i$ takes a value from the half closed interval $(0, 1]$ and $x_i = a + (b - a)y_i$, $x_i$ is documented as taking values from the closed interval $[a,b]$. This is because for some values of $a$ and $b$, nag_rand_uniform (g05sqc) may return a value of $a$ due to numerical rounding.

10 Example

This example prints five pseudorandom numbers from a uniform distribution between $-1.0$ and $1.0$, generated by a single call to nag_rand_uniform (g05sqc), after initialization by nag_rand_init_repeatable (g05kfc).

10.1 Program Text

/* nag_rand_uniform (g05sqc) 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 a = -1.0e0;
    double b = 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_uniform (g05sqc) 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;
    /* Call nag_rand_uniform */
    fail.code = 0;
    x[0] = x[n-1] = 0;
    for (i = 0; i < n-1; i++)
        x[i] = nag_rand_uniform(genid, subid, state, &lstate, &fail);
        if (fail.code != NE_NOERROR)
        {
            printf("Error from nag_rand_uniform (g05sqc).\n%s\n", fail.message);
            exit_status = 1;
        /* Print the results */
        printf("nag_rand_uniform (g05sqc) Example Program Results\n\n");
        printf("Component of the pseudorandom numbers from the\nuniform distribution between -1.0 and 1.0:\n\n");
        for (i = 0; i < n; i++)
            printf("%10.4f\n", x[i]);
        return exit_status;
    return exit_status;
    exit_status = 1;
}
/* 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 a = -1.0e0;
    double b = 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_uniform (g05sqc) 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;
    /* Call nag_rand_uniform */
    fail.code = 0;
    x[0] = x[n-1] = 0;
    for (i = 0; i < n-1; i++)
        x[i] = nag_rand_uniform(genid, subid, state, &lstate, &fail);
        if (fail.code != NE_NOERROR)
        {
            printf("Error from nag_rand_uniform (g05sqc).\n%s\n", fail.message);
            exit_status = 1;
        /* Print the results */
        printf("nag_rand_uniform (g05sqc) Example Program Results\n\n");
        printf("Component of the pseudorandom numbers from the\nuniform distribution between -1.0 and 1.0:\n\n");
        for (i = 0; i < n; i++)
            printf("%10.4f\n", x[i]);
        return exit_status;
    return exit_status;
    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_uniform(n, a, b, state, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_uniform (g05sqc).\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_uniform (g05sqc) Example Program Results

  0.2727
-0.7870
  0.4921
  0.5965
-0.7908