NAG Library Function Document

nag_rand_basic (g05sac)

1 Purpose

nag_rand_basic (g05sac) generates a vector of pseudorandom numbers taken from a uniform distribution between 0 and 1.

2 Specification

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

void nag_rand_basic (Integer n, Integer state[], double x[], NagError *fail)
```

3 Description

nag_rand_basic (g05sac) generates $n$ values from a uniform distribution over the half closed interval $[0, 1]$.

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_basic (g05sac).

4 References


5 Arguments

1:  
   **n** – Integer  
   __Input__  
   
   On entry: $n$, the number of pseudorandom numbers to be generated.  
   
   Constraint: $n \geq 0$.  

2:  
   **state**[$\text{dim}$] – Integer  
   __Communication Array__  
   
   On entry: contains information on the selected base generator and its current state.  
   
   On exit: contains updated information on the state of the generator.

3:  
   **x**[$n$] – double  
   __Output__  
   
   On exit: the $n$ pseudorandom numbers from a uniform distribution over the half closed interval $[0, 1]$.  

4:  
   **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 value \rangle had an illegal value.

NE_INT
On entry, \( n = \langle value \rangle \).
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, \textit{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.

7 Accuracy
Not applicable.

8 Parallelism and Performance
nag\_rand\_basic (g05sac) 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 the first five pseudorandom numbers from a uniform distribution between 0 and 1, generated by nag\_rand\_basic (g05sac) after initialization by nag\_rand\_init\_repeatable (g05kfc).
10.1 Program Text

/* nag_rand_basic (g05sac) 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 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_basic (g05sac) 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;
    }

    END:
    return exit_status;
}
goto END;

/* Generate the variates */
nag_rand_basic(n, state, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_basic (g05sac).\n\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_basic (g05sac) Example Program Results

0.6364
0.1065
0.7460
0.7983
0.1046