1 Purpose

nag_rand_poisson (g05tjc) generates a vector of pseudorandom integers from the discrete Poisson distribution with mean $\lambda$.

2 Specification

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

void nag_rand_poisson (Nag_ModeRNG mode, Integer n, double lambda, 
                        double r[], Integer lr, Integer state[], Integer x[], NagError *fail)
```

3 Description

nag_rand_poisson (g05tjc) generates $n$ integers $x_i$ from a discrete Poisson distribution with mean $\lambda$, where the probability of $x_i = I$ is

$$P(x_i = I) = \frac{\lambda^I \times e^{-\lambda}}{I!}, \quad I = 0, 1, \ldots,$$

where $\lambda \geq 0$.

The variates can be generated with or without using a search table and index. If a search table is used then it is stored with the index in a reference vector and subsequent calls to nag_rand_poisson (g05tjc) with the same parameter values can then use this reference vector to generate further variates. The reference array is found using a recurrence relation if $\lambda$ is less than 50 and by Stirling’s formula otherwise.

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_poisson (g05tjc).

4 References


5 Arguments

1. `mode` – Nag_ModeRNG

   **Input**

   *mode* = Nag_InitializeReference
   Set up reference vector only.

   *mode* = Nag_GenerateFromReference
   Generate variates using reference vector set up in a prior call to nag_rand_poisson (g05tjc).

   *mode* = Nag_InitializeAndGenerate
   Set up reference vector and generate variates.
mode = Nag_GenerateWithoutReference
    Generate variates without using the reference vector.

Constraint: mode = Nag_InitializeReference, Nag_GenerateFromReference,
            Nag_InitializeAndGenerate or Nag_GenerateWithoutReference.

2:  n – Integer
    Input
    On entry: n, the number of pseudorandom numbers to be generated.
    Constraint: n ≥ 0.

3:  lambda – double
    Input
    On entry: λ, the mean of the Poisson distribution.
    Constraint: lambda ≥ 0.0.

4:  r[lr] – double
    Communication Array
    On entry: if mode = Nag_GenerateFromReference, the reference vector from the previous call to
              nag_rand_poisson (g05tjc).
              If mode = Nag_GenerateWithoutReference, r is not referenced and may be NULL.
    On exit: if mode ≠ Nag_GenerateWithoutReference, the reference vector.

5:  lr – Integer
    Input
    On entry: the dimension of the array r.
    Suggested value:
    if mode ≠ Nag_GenerateWithoutReference, lr = 30 + 20 × \sqrt{\lambda} + \lambda;
    otherwise lr = 1.
    Constraints:
    if mode = Nag_InitiaizeReference or Nag_InitializeAndGenerate,
        if \sqrt{\lambda} > 7.15, lr > 9 + \text{int}(8.5 + 14.3 × \sqrt{\lambda});
        otherwise lr > 9 + \text{int}(\lambda + 7.15 × \sqrt{\lambda} + 8.5);.
    if mode = Nag_GenerateFromReference, lr must remain unchanged from the previous call
to nag_rand_poisson (g05tjc).

6:  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 (g05kge).
    On entry: contains information on the selected base generator and its current state.
    On exit: contains updated information on the selected base generator.

7:  x[n] – Integer
    Output
    On exit: the n pseudorandom numbers from the specified Poisson distribution.

8:  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, lr is too small when mode = Nag_InitializeReference or Nag_InitializeAndGenerate:
lr = ⟨value⟩, minimum length required = ⟨value⟩.
On entry, n = ⟨value⟩.
Constraint: n ≥ 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_PREV_CALL
lambda is not the same as when r was set up in a previous call.
Previous value of lambda = ⟨value⟩ and lambda = ⟨value⟩.

NE_REAL
lambda is such that lr would have to be larger than the largest representable integer. Use
mode = Nag_GenerateWithoutReference instead. lambda = ⟨value⟩.
On entry, lambda = ⟨value⟩.
Constraint: lambda ≥ 0.0.

NE_REF_VEC
On entry, some of the elements of the array r have been corrupted or have not been initialized.

7 Accuracy
Not applicable.

8 Parallelism and Performance
nag_rand_poisson (g05tjc) 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 10 pseudorandom integers from a Poisson distribution with mean $\lambda = 20$, generated by a single call to nag_rand_poisson (g05tjc), after initialization by nag_rand_init_repeatable (g05kfc).

10.1 Program Text

/* nag_rand_poisson (g05tjc) 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 lr, i, lstate;
    Integer *state = 0, *x = 0;
    /* NAG structures */
    NagError fail;
    Nag_ModeRNG mode;
    /* Double scalar and array declarations */
    double *r = 0;
    /* Set the distribution parameters */
    double lambda = 20.0e0;
    /* Set the sample size */
    Integer n = 10;
    /* 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_poisson (g05tjc) 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\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\n",

fail.message);
exit_status = 1;
goto END;
}

/* Calculate the size of the reference vector */
lr = 30 + 20 * sqrt(lambda) + lambda;

/* Allocate arrays */
if (!(r = NAG_ALLOC(lr, double)) ||
    !(state = NAG_ALLOC(lstate, Integer)) ||
    !(x = NAG_ALLOC(n, 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, initialising the reference vector
at the same time */
mode = Nag_InitializeAndGenerate;
nag_rand_poisson(mode, n, lambda, r, lr, state, x, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_rand_poisson (g05tjc).\n%s\n", fail.message);
    exit_status = 1;
goto END;
}

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

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

10.2 Program Data

None.

10.3 Program Results

nag_rand_poisson (g05tjc) Example Program Results

21
15
23
24
14

Mark 25