NAG C Library Function Document

nag_rnds_compd_poisson (g05mec)

1 Purpose

nag_rnds_compd_poisson (g05mec) generates a vector of pseudo-random integers, each from a discrete Poisson distribution with differing parameter \( \lambda \).

2 Specification

```c
void nag_rnds_compd_poisson (Integer m, const double vlamda[], Integer x[],
    Integer igen, Integer iseed[], NagError *fail)
```

3 Description

nag_rnds_compd_poisson (g05mec) generates \( m \) integers \( x_j \), each from a discrete Poisson distribution with mean \( \lambda_j \), where the probability of \( x_j = I \) is

\[
P(x_j = I) = \frac{\lambda_j^I e^{-\lambda_j}}{I!}, \quad I = 0, 1, \ldots,
\]

where

\[
0 \leq \lambda_j, \quad j = 1, 2, \ldots, m.
\]

The methods used by this function have low set up times and are designed for efficient use when the value of the parameter \( \lambda \) changes during the simulation. For large samples from a distribution with fixed \( \lambda \) using nag_rnds_poisson (g05mkc) to set up and use a reference vector may be more efficient.

When \( \lambda < 7.5 \) the product of uniforms method is used, see for example Dagpunar (1988). For larger values of \( \lambda \) an envelope rejection method is used with a target distribution:

\[
f(x) = \begin{cases} \frac{1}{7} & \text{if } |x| \leq 1, \\ \frac{1}{12} |x|^{-1} & \text{otherwise}. \end{cases}
\]

This distribution is generated using a ratio of uniforms method. A similar approach has also been suggested by Ahrens and Dieter (1989). The basic method is combined with quick acceptance and rejection tests given by Maclaren (1990). For values of \( \lambda \geq 87 \) Stirling’s approximation is used in the computation of the Poisson distribution function, otherwise tables of factorials are used as suggested by Maclaren (1990).

One of the initialisation functions nag_rnds_init_repeatable (g05kbc) (for a repeatable sequence if computed sequentially) or nag_rnds_init_nonrepeatable (g05kcc) (for a non-repeatable sequence) must be called prior to the first call to nag_rnds_compd_poisson (g05mec).

4 References


Maclaren N M (1990) A Poisson random number generator Personal Communication
5 Parameters

1: \( m \) – Integer

\textit{Input}

On entry: the number, \( m \), of Poisson distributions for which pseudo-random variates are required.

\textit{Constraint:} \( m \geq 1 \).

2: \( \text{vlamda}[m] \) – const double

\textit{Input}

On entry: the means, \( \lambda_j \), for \( j = 1, 2, \ldots, m \), of the Poisson distributions.

\textit{Constraint:} \( 0.0 \leq \text{vlamda}[j] \leq \text{maxint}/2 \), where \( \text{maxint} \) is the largest integer representable on the machine (see \text{nag_max_integer (X02BBC)}).

3: \( x[m] \) – Integer

\textit{Output}

On exit: the \( m \) pseudo-random numbers from the specified Poisson distributions.

4: \( igen \) – Integer

\textit{Input}

On entry: must contain the identification number for the generator to be used to return a pseudo-random number and should remain unchanged following initialisation by a prior call to one of the functions \text{nag_rngs_init_repeatable (g05kbc)} or \text{nag_rngs_init_nonrepeatable (g05kcc)}.

5: \( \text{iseed}[4] \) – Integer

\textit{Input/Output}

On entry: contains values which define the current state of the selected generator.

On exit: contains updated values defining the new state of the selected generator.

6: \( \text{fail} \) – NagError *

\textit{Input/Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_INT}

On entry, \( m = \langle \text{value} \rangle \).

\textit{Constraint:} \( m \geq 1 \).

\textbf{NE_REAL_ARRAY_ELEM_CONS}

On entry, \( 2 \times \text{vlamda}[i - 1] > \text{maxint} \) for at least one value of \( i = 1, 2, \ldots, m \).

On entry, at least one element of \( \text{vlamda} < 0.0 \).

\textbf{NE_BAD_PARAM}

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.

\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 consult NAG for assistance.

7 Accuracy

Not applicable.

8 Further Comments

None.
9 Example

The example program prints ten pseudo-random integers from five Poisson distributions with means \( \lambda_1 = 0.5, \lambda_2 = 5, \lambda_3 = 10, \lambda_4 = 50 \) and \( \lambda_5 = 100 \). These are generated by ten calls to \texttt{nag_rngs_compd_poisson(g05mec)}, after initialisation by \texttt{nag_rngs_init_repeatable(g05kbc)}.

9.1 Program Text

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

int main(void)
{
    /* Scalars */
    Integer i, igen, j, m, n;
    Integer exit_status=0;
    NagError fail;

    /* Arrays */
    double *vlamda=0;
    Integer *x=0;
    Integer iseed[4];

    INIT_FAIL(fail);
    Vprintf("g05mec Example Program Results\n\n");
    m = 5;
    n = 10;

    /* Allocate memory */
    if ( !(vlamda = NAG_ALLOC(m, double)) ||
        !(x = NAG_ALLOC(m, Integer)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Set the distribution parameter LAMBDA */
    vlamda[0] = 0.5;
    vlamda[1] = 5.0;
    vlamda[2] = 10.0;
    vlamda[3] = 500.0;
    vlamda[4] = 1e3;

    /* Initialise the seed to a repeatable sequence */
    iseed[0] = 1762543;
    iseed[1] = 9324783;
    iseed[2] = 423442;
    iseed[3] = 742355;

    igen = 1;
    g05kbc(&igen, iseed);

    /* Generate integers and store in X */
    for (i = 0; i < n; ++i)
    {
        g05mec(m, vlamda, x, igen, iseed, &fail);
        if (fail.code != NE_NOERROR)
        {
            Vprintf("Error from g05mec.\n\n", fail.message);
            exit_status = 1;
        }
    }
}
```

[g05 – Random Number Generators

\texttt{g05mec}.

[NP3645/7]
goto END;
}
for (j = 0; j < m; ++j)
{
    Vprintf("%12ld%s", x[j], (j+1)%5 == 0 || j == 4 ?"\n":" ");
}
}
END:
if (vlambda) NAG_FREE(vlambda);
if (x) NAG_FREE(x);
return exit_status;

9.2 Program Data

None.

9.3 Program Results

g05mec Example Program Results

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>13</td>
<td>482</td>
<td>1001</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>12</td>
<td>494</td>
<td>1046</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>11</td>
<td>548</td>
<td>941</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>8</td>
<td>518</td>
<td>977</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>8</td>
<td>504</td>
<td>943</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>8</td>
<td>502</td>
<td>991</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>7</td>
<td>475</td>
<td>991</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>507</td>
<td>1012</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>13</td>
<td>537</td>
<td>1016</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>7</td>
<td>492</td>
<td>1072</td>
</tr>
</tbody>
</table>