NAG Toolbox

nag_stat_pdf_gamma_vector (g01kk)

1 Purpose

nag_stat_pdf_gamma_vector (g01kk) returns a number of values of the probability density function (PDF), or its logarithm, for the gamma distribution.

2 Syntax

```
[pdf, ivalid, ifail] = nag_stat_pdf_gamma_vector(ilog, x, a, b, 'lx', lx, 'la', la, 'lb', lb)
[pdf, ivalid, ifail] = g01kk(ilog, x, a, b, 'lx', lx, 'la', la, 'lb', lb)
```

3 Description

The gamma distribution with shape parameter α_i and scale parameter β_i has PDF

$$f(x_i,\alpha_i,\beta_i) = \frac{1}{{\beta_i}^{\alpha_i}\Gamma(\alpha_i)}{x_i}^{\alpha_i-1}e^{-x_i/\beta_i} \quad \text{if } x_i \geq 0; \quad \alpha_i,\beta_i > 0$$

$$f(x_i,\alpha_i,\beta_i) = 0 \qquad \text{otherwise.}$$

If $0.01 \leq x_i,\alpha_i,\beta_i \leq 100$ then an algorithm based directly on the gamma distribution's PDF is used. For values outside this range, the function is calculated via the Poisson distribution's PDF as described in Loader (2000) (see Section 9).

The input arrays to this function are designed to allow maximum flexibility in the supply of vector arguments by re-using elements of any arrays that are shorter than the total number of evaluations required. See Section 2.6 in the G01 Chapter Introduction for further information.

4 References

Loader C (2000) Fast and accurate computation of binomial probabilities **(not yet published)**

5 Parameters

5.1 Compulsory Input Parameters

1: **ilog** – INTEGER

The value of **ilog** determines whether the logarithmic value is returned in **pdf**.

ilog $= 0$
$f(x_i,\alpha_i,\beta_i)$, the probability density function is returned.

ilog $= 1$
$\log(f(x_i,\alpha_i,\beta_i))$, the logarithm of the probability density function is returned.

Constraint: **ilog** $= 0$ or 1.

2: **x**(**lx**) – REAL (KIND=nag_wp) array

x_i, the values at which the PDF is to be evaluated with $x_i = \mathbf{x}(j)$, $j = ((i-1) \bmod \mathbf{lx}) + 1$, for $i = 1,2,\ldots,\max(\mathbf{lx},\mathbf{la},\mathbf{lb})$.

3: **a**(**la**) – REAL (KIND=nag_wp) array

α_i, the shape parameter with $\alpha_i = \mathbf{a}(j)$, $j = ((i-1) \bmod \mathbf{la}) + 1$.

Constraint: $\mathbf{a}(j) > 0.0$, for $j = 1, 2, \ldots, \mathbf{la}$.

4: **b**(**lb**) – REAL (KIND=nag_wp) array

β_i, the scale parameter with $\beta_i = \mathbf{b}(j)$, $j = ((i-1) \bmod \mathbf{lb}) + 1$.

Constraint: $\mathbf{b}(j) > 0.0$, for $j = 1, 2, \ldots, \mathbf{lb}$.

5.2 Optional Input Parameters

1: **lx** – INTEGER

Default: the dimension of the array **x**.

The length of the array **x**.

Constraint: $\mathbf{lx} > 0$.

2: **la** – INTEGER

Default: the dimension of the array **a**.

The length of the array **a**.

Constraint: $\mathbf{la} > 0$.

3: **lb** – INTEGER

Default: the dimension of the array **b**.

The length of the array **b**.

Constraint: $\mathbf{lb} > 0$.

5.3 Output Parameters

1: **pdf**(:) – REAL (KIND=nag_wp) array

The dimension of the array **pdf** will be $\max(\mathbf{lx}, \mathbf{la}, \mathbf{lb})$

$f(x_i, \alpha_i, \beta_i)$ or $\log(f(x_i, \alpha_i, \beta_i))$.

2: **ivalid**(:) – INTEGER array

The dimension of the array **ivalid** will be $\max(\mathbf{lx}, \mathbf{la}, \mathbf{lb})$

ivalid(i) indicates any errors with the input arguments, with

ivalid$(i) = 0$
No error.

ivalid$(i) = 1$
$\alpha_i \leq 0.0$.

ivalid$(i) = 2$
$\beta_i \leq 0.0$.

ivalid$(i) = 3$
$\frac{x_i}{\beta_i}$ overflows, the value returned should be a reasonable approximation.

3: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$ (*warning*)

On entry, at least one value of **x**, **a** or **b** was invalid.
Check **ivalid** for more information.

ifail $= 2$

Constraint: **ilog** $= 0$ or 1.

ifail $= 3$

Constraint: **lx** > 0.

ifail $= 4$

Constraint: **la** > 0.

ifail $= 5$

Constraint: **lb** > 0.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

Due to the lack of a stable link to Loader (2000) paper, we give a brief overview of the method, as applied to the Poisson distribution. The Poisson distribution has a continuous mass function given by,

$$p(x;\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}. \tag{1}$$

The usual way of computing this quantity would be to take the logarithm and calculate,

$$\log\left(p(x;\lambda)\right) = x\log\lambda - \log\left(x!\right) - \lambda.$$

For large x and λ, $x\log\lambda$ and $\log\left(x!\right)$ are very large, of the same order of magnitude and when calculated have rounding errors. The subtraction of these two terms can therefore result in a number, many orders of magnitude smaller and hence we lose accuracy due to subtraction errors. For example for $x = 2 \times 10^6$ and $\lambda = 2 \times 10^6$, $\log\left(x!\right) \approx 2.7 \times 10^7$ and $\log\left(p(x;\lambda)\right) = -8.17326744645834$. But calculated with the method shown later we have $\log\left(p(x;\lambda)\right) = -8.1732674441334492$. The difference between these two results suggests a loss of about 7 significant figures of precision.

Loader introduces an alternative way of expressing (1) based on the saddle point expansion,

$$\log\left(p(x;\lambda)\right) = \log\left(p(x;x)\right) - D(x;\lambda), \tag{2}$$

where $D(x;\lambda)$, the deviance for the Poisson distribution is given by,

$$\begin{aligned} D(x;\lambda) &= \log\left(p(x;x)\right) - \log\left(p(x;\lambda)\right), \\ &= \lambda D_0\left(\tfrac{x}{\lambda}\right), \end{aligned} \tag{3}$$

and

$$D_0(\epsilon) = \epsilon\log\epsilon + 1 - \epsilon.$$

For ϵ close to 1, $D_0(\epsilon)$ can be evaluated through the series expansion

$$\lambda D_0\left(\frac{x}{\lambda}\right) = \frac{(x-\lambda)^2}{x+\lambda} + 2x\sum_{j=1}^{\infty}\frac{v^{2j+1}}{2j+1}, \qquad \text{where } v = \frac{x-\lambda}{x+\lambda},$$

otherwise $D_0(\epsilon)$ can be evaluated directly. In addition, Loader suggests evaluating $\log(x!)$ using the Stirling–De Moivre series,

$$\log(x!) = \frac{1}{2}\log(2\pi x) + x\log(x) - x + \delta(x), \tag{4}$$

where the error $\delta(x)$ is given by

$$\delta(x) = \frac{1}{12x} - \frac{1}{360x^3} + \frac{1}{1260x^5} + \mathcal{O}\left(x^{-7}\right).$$

Finally $\log\left(p(x;\lambda)\right)$ can be evaluated by combining equations (1)–(4) to get,

$$p(x;\lambda) = \frac{1}{\sqrt{2\pi x}}e^{-\delta(x)-\lambda D_0(x/\lambda)}.$$

9 Example

This example prints the value of the gamma distribution PDF at six different points x_i with differing α_i and β_i.

9.1 Program Text

```
     function g01kk_example

fprintf('g01kk example results\n\n');

x = [0.1,  3,  6,  4,    9,   16  ];
a = [3,   10,  5, 10,    9,    3.5];
b = [2,   11,  1,  0.1,  0.5,  2.5];

ilog = nag_int(0);
[pdf, ivalid, ifail] = g01kk( ...
       ilog, x, a, b);

fprintf('     x       a       b       pdf     ivalid\n');
lx = numel(x);
la = numel(a);
lb = numel(b);
len = max ([lx, la, lb]);

for i=0:len-1
  fprintf('%8.2f%8.2f%8.2f%12.4e%4d\n', x(mod(i,lx)+1), a(mod(i,la)+1), ...
   b(mod(i,lb)+1), pdf(i+1), ivalid(i+1));
end
```

9.2 Program Results

```
     g01kk example results

      x       a       b       pdf     ivalid
   0.10    3.00    2.00  5.9452e-04   0
   3.00   10.00   11.00  1.5921e-12   0
   6.00    5.00    1.00  1.3385e-01   0
   4.00   10.00    0.10  3.0690e-08   0
   9.00    9.00    0.50  8.3251e-03   0
  16.00    3.50    2.50  2.0723e-02   0
```