NAG C Library Function Document

nag_prob_lin_non_central_chi_sq (g01jcc)

1 Purpose

nag_prob_lin_non_central_chi_sq (g01jcc) returns the lower tail probability of a distribution of a positive linear combination of χ^2 random variables.

2 Specification

```
void nag_prob_lin_non_central_chi_sq (const double a[], const Integer mult[],
     const double rlamda[], Integer n, double c, double *p, double *pdf, double tol,
     Integer maxit, NagError *fail)
```

3 Description

For a linear combination of non-central χ^2 random variables with integer degrees of freedom the lower tail probability is

$$P\left(\sum_{j=1}^{n} a_j\chi^2(m_j,\lambda_j) \leq c\right), \tag{1}$$

where a_j and c are positive constants and where $\chi^2(m_j,\lambda_j)$ represents an independent χ^2 random variable with m_j degrees of freedom and non-centrality parameter λ_j. The linear combination may arise from considering a quadratic form in Normal variables.

Ruben's method as described in Farebrother (1984) is used. Ruben has shown that (1) may be expanded as an infinite series of the form

$$\sum_{k=0}^{\infty} d_k F(m+2k, c/\beta), \tag{2}$$

where $F(m+2k,c/\beta) = P\big(\chi^2(m+2k) < c/\beta\big)$, i.e., the probability that a central χ^2 is less than c/β.

The value of β is set at

$$\beta = \beta_B = \frac{2}{(1/a_{\mathrm{min}} + 1/a_{\mathrm{max}})}$$

unless $\beta_B > 1.8a_{\mathrm{min}}$, in which case

$$\beta = \beta_A = a_{\mathrm{min}}$$

is used, where $a_{\mathrm{min}} = \min\{a_j\}$ and $a_{\mathrm{max}} = \max\{a_j\}$, for $j = 1,2,\ldots,n$.

4 References

Farebrother R W (1984) The distribution of a positive linear combination of χ^2 random variables *Appl. Statist.* **33 (3)**

5 Parameters

1: **a**[**n**] – const double *Input*

On entry: the weights, $a_1, a_2, \ldots, a_n$.

Constraint: **a**$[i] > 0.0$ for $i = 0,1,\ldots,n-1$.

2: **mult**[**n**] – const Integer *Input*

On entry: the degrees of freedom, $m_1, m_2, \ldots, m_n$.

Constraint: **mult**$[i] \geq 1$ for $i = 0, 1, \ldots, \mathbf{n} - 1$.

3: **rlamda**[**n**] – const double *Input*

On entry: the non-centrality parameters, $\lambda_1, \lambda_2, \ldots, \lambda_n$.

Constraint: **rlamda**$[i] \geq 0.0$ for $i = 0, 1, \ldots, n - 1$.

4: **n** – Integer *Input*

On entry: the number of χ^2 random variables in the combination, n, i.e., the number of terms in equation (1).

Constraint: $\mathbf{n} \geq 1$.

5: **c** – double *Input*

On entry: the point for which the lower tail probability is to be evaluated, c.

Constraint: $\mathbf{c} \geq 0.0$.

6: **p** – double * *Output*

On exit: the lower tail probability associated with the linear combination of n χ^2 random variables with m_j degrees of freedom, and non-centrality parameters λ_j, for $j = 1, 2, \ldots, n$.

7: **pdf** – double * *Output*

On exit: the value of the probability density function of the linear combination of χ^2 variables.

8: **tol** – double *Input*

On entry: the relative accuracy required by the user in the results. If nag_prob_lin_non_central_chi_sq (g01jcc) is entered with **tol** greater than or equal to 1.0 or less than 10 times the ***machine precision*** (see nag_machine_precision (X02AJC)), then the value of 10 times ***machine precision*** is used instead.

9: **maxit** – Integer *Input*

On entry: the maximum number of terms that should be used during the summation.

Suggested value: 500.

Constraint: $\mathbf{maxit} \geq 1$.

10: **fail** – NagError * *Input/Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **maxit** = ⟨*value*⟩.
Constraint: $\mathbf{maxit} \geq 1$.

On entry, **n** = ⟨*value*⟩.
Constraint: $\mathbf{n} \geq 1$.

NE_ACCURACY

The required accuracy could not be met in ⟨*value*⟩ iterations.

NE_CONVERGENCE

The central Chi square has failed to converge.

NE_PROB_BOUNDARY

Calculated probability at boundary.

NE_REAL

On entry, **c** = ⟨*value*⟩.
Constraint: **c** $\geq$ 0.0.

NE_REAL_ARRAY

On entry, **rlamda** has an element $<$ 0.0: **rlamda**[⟨*value*⟩] = ⟨*value*⟩.

On entry, **a** has an element $\leq$ 0.0: **a**[⟨*value*⟩] = ⟨*value*⟩.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ had an illegal value.

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

The series (2) is summed until a bound on the truncation error is less than **tol**. See Farebrother (1984) for further discussion.

8 Further Comments

None.

9 Example

The number of χ^2 variables is read along with their coefficients, degrees of freedom and non-centrality parameters. The lower tail probability is then computed and printed.

9.1 Program Text

```
/* nag_prob_lin_non_central_chi_sq (g01jcc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{
  /* Initialized data */
  Integer maxit = 500;
  double tol = 1e-4;
```

```
  /* Scalars */
  double c, p, pdf;
  Integer exit_status, i, n;

  NagError fail;

  /* Arrays */
  double *a=0, *rlamda=0;
  Integer *mult=0;

  INIT_FAIL(fail);
  exit_status = 0;
  Vprintf("g01jcc Example Program Results\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");

  Vprintf("\n        A     MULT  RLAMDA\n");
  while (scanf("%ld%lf%*[^\n] ", &n, &c) != EOF)
    {
      /* Allocate memory */
      if ( !(a = NAG_ALLOC(n, double)) ||
           !(rlamda = NAG_ALLOC(n, double)) ||
           !(mult = NAG_ALLOC(n, Integer)) )
        {
          Vprintf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      Vprintf("\n");
      for (i = 1; i <= n; ++i)
        Vscanf("%lf", &a[i - 1]);
      Vscanf("%*[^\n] ");

      for (i = 1; i <= n; ++i)
        Vscanf("%ld", &mult[i - 1]);
      Vscanf("%*[^\n] ");
      for (i = 1; i <= n; ++i)
        Vscanf("%lf", &rlamda[i - 1]);
      Vscanf("%*[^\n] ");

      g01jcc(a, mult, rlamda, n, c, &p, &pdf, tol, maxit, &fail);
      if (fail.code == NE_NOERROR || fail.code == NE_ACCURACY ||
          fail.code == NE_PROB_BOUNDARY)
        {
          for (i = 1; i <= n; ++i)
            Vprintf(" %10.2f%6ld%9.2f\n", a[i - 1], mult[i - 1], rlamda[i - 1]);
          Vprintf("c = %6.2f    Prob = %6.4f\n", c, p);
        }
      else
        {
          Vprintf("Error from g01dac.\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }

      if (a) NAG_FREE(a);
      if (rlamda) NAG_FREE(rlamda);
      if (mult) NAG_FREE(mult);
    }

 END:
  if (a) NAG_FREE(a);
  if (rlamda) NAG_FREE(rlamda);
  if (mult) NAG_FREE(mult);
  return exit_status;
}
```

9.2 Program Data

```
g01jcc Example Program Data
 3    20.0                        :N  C
 6.0   3.0  1.0                   :A(I), I=1,N
 1     1    1                     :MULT(I), I=1,N
 0.0   0.0  0.0                   :RLAMDA(I), I=1,N
 2    10.0                        :N  C
 7.0   3.0                        :A(I), I=1,N
 1     1                          :MULT(I), I=1,N
 6.0   2.0                        :RLAMDA(I), I=1,N
```

9.3 Program Results

```
g01jcc Example Program Results

        A      MULT  RLAMDA

       6.00     1     0.00
       3.00     1     0.00
       1.00     1     0.00
c =  20.00    Prob = 0.8760

       7.00     1     6.00
       3.00     1     2.00
c =  10.00    Prob = 0.0451
```