NAG Library Function Document

nag_prob_lin_chi_sq (g01jdc)

1 Purpose

nag_prob_lin_chi_sq (g01jdc) calculates the lower tail probability for a linear combination of (central) χ^2 variables.

2 Specification

```
#include <nag.h>
#include <nagg01.h>
void nag_prob_lin_chi_sq (Nag_LCCMethod method, Integer n,
     const double rlam[], double d, double c, double *prob, NagError *fail)
```

3 Description

Let $u_1, u_2, \ldots, u_n$ be independent Normal variables with mean zero and unit variance, so that $u_1^2, u_2^2, \ldots, u_n^2$ have independent χ^2-distributions with unit degrees of freedom. nag_prob_lin_chi_sq (g01jdc) evaluates the probability that

$$\lambda_1 u_1^2 + \lambda_2 u_2^2 + \cdots + \lambda_n u_n^2 < d\left(u_1^2 + u_2^2 + \cdots + u_n^2\right) + c.$$

If $c = 0.0$ this is equivalent to the probability that

$$\frac{\lambda_1 u_1^2 + \lambda_2 u_2^2 + \cdots + \lambda_n u_n^2}{u_1^2 + u_2^2 + \cdots + u_n^2} < d.$$

Alternatively let

$$\lambda_i^* = \lambda_i - d, \qquad i = 1, 2, \ldots, n,$$

then nag_prob_lin_chi_sq (g01jdc) returns the probability that

$$\lambda_1^* u_1^2 + \lambda_2^* u_2^2 + \cdots + \lambda_n^* u_n^2 < c.$$

Two methods are available. One due to Pan (1964) (see Farebrother (1980)) makes use of series approximations. The other method due to Imhof (1961) reduces the problem to a one-dimensional integral. If $n \geq 6$ then a non-adaptive method is used to compute the value of the integral otherwise nag_1d_quad_gen_1 (d01sjc) is used.

Pan's procedure can only be used if the λ_i^* are sufficiently distinct; nag_prob_lin_chi_sq (g01jdc) requires the λ_i^* to be at least 1% distinct; see Section 9. If the λ_i^* are at least 1% distinct and $n \leq 60$, then Pan's procedure is recommended; otherwise Imhof's procedure is recommended.

4 References

Farebrother R W (1980) Algorithm AS 153. Pan's procedure for the tail probabilities of the Durbin–Watson statistic *Appl. Statist.* **29** 224–227

Imhof J P (1961) Computing the distribution of quadratic forms in Normal variables *Biometrika* **48** 419–426

Pan Jie–Jian (1964) Distributions of the noncircular serial correlation coefficients *Shuxue Jinzhan* **7** 328–337

5 Arguments

1: **method** – Nag_LCCMethod *Input*

On entry: indicates whether Pan's, Imhof's or an appropriately selected procedure is to be used.

method = Nag_LCCPan
Pan's method is used.

method = Nag_LCCImhof
Imhof's method is used.

method = Nag_LCCDefault
Pan's method is used if λ_i^*, for $i = 1, 2, \ldots, n$ are at least 1% distinct and $n \leq 60$; otherwise Imhof's method is used.

Constraint: **method** = Nag_LCCPan, Nag_LCCImhof or Nag_LCCDefault.

2: **n** – Integer *Input*

On entry: n, the number of independent standard Normal variates, (central χ^2 variates).

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

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

On entry: the weights, λ_i, for $i = 1, 2, \ldots, n$, of the central χ^2 variables.

Constraint: $\mathbf{rlam}[i-1] \neq \mathbf{d}$ for at least one i. If **method** = Nag_LCCPan, then the λ_i^* must be at least 1% distinct; see Section 9, for $i = 1, 2, \ldots, n$.

4: **d** – double *Input*

On entry: d, the multiplier of the central χ^2 variables.

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

5: **c** – double *Input*

On entry: c, the value of the constant.

6: **prob** – double * *Output*

On exit: the lower tail probability for the linear combination of central χ^2 variables.

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

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_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_REAL

On entry, **d** $= \langle value \rangle$.
Constraint: **d** ≥ 0.0.

NE_REAL_ARRAY

On entry, all values of **rlam** $=$ **d**.

NE_REAL_ARRAY_ENUM

On entry, **method** $=$ Nag_LCCPan but two successive values of $\lambda*$ were not 1 percent distinct.

7 Accuracy

On successful exit at least four decimal places of accuracy should be achieved.

8 Parallelism and Performance

nag_prob_lin_chi_sq (g01jdc) 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

Pan's procedure can only work if the λ_i^* are sufficiently distinct. nag_prob_lin_chi_sq (g01jdc) uses the check $\left|w_j - w_{j-1}\right| \geq 0.01 \times \max\left(\left|w_j\right|, \left|w_{j-1}\right|\right)$, where the w_j are the ordered nonzero values of λ_i^*.

For the situation when all the λ_i are positive nag_prob_lin_non_central_chi_sq (g01jcc) may be used. If the probabilities required are for the Durbin–Watson test, then the bounds for the probabilities are given by nag_prob_durbin_watson (g01epc).

10 Example

For $n = 10$, the choice of method, values of c and d and the λ_i are input and the probabilities computed and printed.

10.1 Program Text

```
/* nag_prob_lin_chi_sq (g01jdc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
```

```
#include <string.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagg01.h>

int main(void)
{
  /* Scalars */
  double         c, d, prob;
  Integer        exit_status, i, n;
  /* Arrays */
  char           nag_enum_arg[40];
  double         *rlam = 0;
  Nag_LCCMethod method;
  NagError       fail;

  INIT_FAIL(fail);

  exit_status = 0;
  printf("nag_prob_lin_chi_sq (g01jdc) Example Program Results\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
  n = 10;

  /* Allocate memory */
  if (!(rlam = NAG_ALLOC(n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

#ifdef _WIN32
  scanf_s(" %39s%lf%lf%*[^\n] ", nag_enum_arg, _countof(nag_enum_arg), &d, &c);
#else
  scanf(" %39s%lf%lf%*[^\n] ", nag_enum_arg, &d, &c);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  method = (Nag_LCCMethod) nag_enum_name_to_value(nag_enum_arg);

  for (i = 1; i <= n; ++i)
#ifdef _WIN32
    scanf_s("%lf", &rlam[i - 1]);
#else
    scanf("%lf", &rlam[i - 1]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif

  /* nag_prob_lin_chi_sq (g01jdc).
   * Computes lower tail probability for a linear combination
   * of (central) chi^2 variables
   */
  nag_prob_lin_chi_sq(method, n, rlam, d, c, &prob, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_prob_lin_chi_sq (g01jdc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
```

```
  printf("\n");
  printf(" Values of lambda ");
  for (i = 1; i <= n; ++i)
    {
      if (i%6 == 0)
        printf("%18s", " ");

      printf("%6.2f ", rlam[i - 1]);

      if (i%5 == 0 || i == n)
          printf("\n");
    }
  printf(" Value of D       %6.2f\n", d);
  printf(" Value of C       %6.2f\n\n", c);
  printf(" Probability = %11.4f\n", prob);

 END:
  NAG_FREE(rlam);

  return exit_status;
}
```

10.2 Program Data

```
nag_prob_lin_chi_sq (g01jdc) Example Program Data
 Nag_LCCPan 1.0 0.0
 -9.0 -7.0 -5.0 -3.0 -1.0 2.0 4.0 6.0 8.0 10.0
```

10.3 Program Results

```
nag_prob_lin_chi_sq (g01jdc) Example Program Results

 Values of lambda  -9.00  -7.00  -5.00  -3.00  -1.00
                    2.00   4.00   6.00   8.00  10.00
 Value of D         1.00
 Value of C         0.00

 Probability =      0.5749
```