

## NAG C 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)  $\chi^2$  variables.

#### 2 Specification

```
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  $u_1, u_2, \dots, u_n$  be independent Normal variables with mean zero and unit variance, so that  $u_1^2, u_2^2, \dots, u_n^2$  have independent  $\chi^2$  distributions with unit degrees of freedom. nag\_prob\_lin\_chi\_sq (g01jdc) evaluates the probability that

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

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

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

Alternatively let

$$\lambda_i^* = \lambda_i - d, \quad \text{for } i = 1, 2, \dots, n,$$

then nag\_prob\_lin\_chi\_sq (g01jdc) returns the probability that

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

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 (d01ajc) is used.

Pan's procedure can only be used if the  $\lambda_i^*$  are sufficiently distinct; nag\_prob\_lin\_chi\_sq (g01jdc) requires the  $\lambda_i^*$  to be at least 1% distinct; see Section 8. If the  $\lambda_i^*$  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 Parameters

- 1: **method** – Nag\_LCCMethod *Input*  
*On entry:* indicates whether Pan's, Imhof's or an appropriately selected procedure is to be used.  
 If **method** = **Nag.LCCPan**, then Pan's method is used.

If **method** = **Nag\_LCCImhof**, then Imhof's method is used.

If **method** = **Nag\_LCCDefault**, then Pan's method is used if  $\lambda_i^*$ , for  $i = 1, 2, \dots, n$  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:* the number of independent standard Normal variates, (central  $\chi^2$  variates),  $n$ .  
*Constraint:*  $n \geq 1$ .
- 3: **rlam**[**n**] – const double *Input*  
*On entry:* the weights,  $\lambda_i$ , for  $i = 1, 2, \dots, n$ , of the central  $\chi^2$  variables.  
*Constraint:* **rlam**[ $i - 1$ ]  $\neq$  **d** for at least one  $i$ . If **method** = **Nag\_LCCPan**, then the  $\lambda_i^*$ , for  $i = 1, 2, \dots, n$ , must be at least 1% distinct; see Section 8.
- 4: **d** – double *Input*  
*On entry:* the multiplier of the central  $\chi^2$  variables,  $d$ .  
*Constraint:*  $d \geq 0.0$ .
- 5: **c** – double *Input*  
*On entry:* the value of the constant,  $c$ .
- 6: **prob** – double \* *Output*  
*On exit:* the lower tail probability for the linear combination of central  $\chi^2$  variables.
- 7: **fail** – NagError \* *Input/Output*  
The NAG error parameter (see the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_INT

On entry, **n** =  $\langle value \rangle$ .  
*Constraint:*  $n \geq 1$ .

### NE\_REAL

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

### 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.

### NE\_ALLOC\_FAIL

Memory allocation failed.

### NE\_BAD\_PARAM

On entry, parameter  $\langle value \rangle$  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**

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

**8 Further Comments**

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

For the situation when all the  $\lambda_i$  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).

**9 Example**

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

**9.1 Program Text**

```
/* nag_prob_lin_chi_sq (g01jdc) Example Program.
 *
 * Copyright 2001 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 *method=0;
    double *rlam=0;

    Nag_LCCMethod method_enum;
    NagError fail;

    INIT_FAIL(fail);
    exit_status = 0;
    Vprintf("g01jdc Example Program Results\n");

    /* Skip heading in data file */
    Vscanf("%s[^\n] ");
    n = 10;

    /* Allocate memory */
    if ( !(method = NAG_ALLOC(2, char)) ||
        !(rlam = NAG_ALLOC(n, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
    }
}
```

```

        goto END;
    }

    Vscanf(" ' %ls '%lf%lf%*[^\\n] ", method, &d, &c);
    for (i = 1; i <= n; ++i)
        Vscanf("%lf", &rlam[i - 1]);
    Vscanf("%*[^\\n] ");

    if (!(strcmp(method, "P")))
        method_enum = Nag_LCCPan;
    else if (!(strcmp(method, "I")))
        method_enum = Nag_LCCImhof;
    else if (!(strcmp(method, "D")))
        method_enum = Nag_LCCDefault;
    else
    {
        Vprintf("The character 'method' is invalid\\n");
        exit_status = -1;
        goto END;
    }

    g01jdc(method_enum, n, rlam, d, c, &prob, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from g01jdc.\\n%s\\n", fail.message);
        exit_status = 1;
        goto END;
    }

    Vprintf("\\n");
    Vprintf("Values of lambda ");
    for (i = 1; i <= n; ++i)
    {
        Vprintf("%6.2f", rlam[i - 1]);

        Vprintf(i%10 == 0 || i == 10 ? "\\n":" ");
    }
    Vprintf(" Value of D      %6.2f\\n", d);
    Vprintf(" Value of C      %6.2f\\n\\n", c);
    Vprintf(" Probability = %10.4f\\n", prob);

END:
    if (method) NAG_FREE(method);
    if (rlam) NAG_FREE(rlam);

    return exit_status;
}

```

## 9.2 Program Data

g01jdc Example Program Data  
 'P' 1.0 0.0  
 -9.0 -7.0 -5.0 -3.0 -1.0 2.0 4.0 6.0 8.0 10.0

## 9.3 Program Results

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

```

---