NAG Library Routine Document

G01SDF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G01SDF returns a number of lower or upper tail probabilities for the F or variance-ratio distribution with real degrees of freedom.

2 Specification

```
SUBROUTINE G01SDF (LTAIL, TAIL, LF, F, LDF1, DF1, LDF2, DF2, P, IVALID,    &
                   IFAIL)

INTEGER             LTAIL, LF, LDF1, LDF2, IVALID(*), IFAIL
REAL (KIND=nag_wp) F(LF), DF1(LDF1), DF2(LDF2), P(*)
CHARACTER(1)        TAIL(LTAIL)
```

3 Description

The lower tail probability for the F, or variance-ratio, distribution with u_i and v_i degrees of freedom, $P\left(F_i \leq f_i : u_i, v_i\right)$, is defined by:

$$P\left(F_i \leq f_i : u_i, v_i\right) = \frac{{u_i}^{u_i/2}{v_i}^{v_i/2}\Gamma((u_i+v_i)/2)}{\Gamma(u_i/2)\Gamma(v_i/2)}\int_0^{f_i} {F_i}^{(u_i-2)/2}\left(u_iF_i+v_i\right)^{-(u_i+v_i)/2}dF_i,$$

for $u_i,\ v_i > 0,\ f_i \geq 0$.

The probability is computed by means of a transformation to a beta distribution, $P_{\beta_i}\left(B_i \leq \beta_i : a_i, b_i\right)$:

$$P\left(F_i \leq f_i : u_i, v_i\right) = P_{\beta_i}\left(B_i \leq \frac{u_if_i}{u_if_i+v_i} : u_i/2, v_i/2\right)$$

and using a call to G01EEF.

For very large values of both u_i and v_i, greater than 10^5, a normal approximation is used. If only one of u_i or v_i is greater than 10^5 then a χ^2 approximation is used, see Abramowitz and Stegun (1972).

The input arrays to this routine 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

Abramowitz M and Stegun I A (1972) *Handbook of Mathematical Functions* (3rd Edition) Dover Publications

Hastings N A J and Peacock J B (1975) *Statistical Distributions* Butterworth

5 Arguments

1: LTAIL – INTEGER *Input*

On entry: the length of the array TAIL.

Constraint: LTAIL > 0.

2: TAIL(LTAIL) – CHARACTER(1) array *Input*

On entry: indicates whether the lower or upper tail probabilities are required. For $j = ((i-1) \bmod \text{LTAIL}) + 1$, for $i = 1, 2, \ldots, \max(\text{LTAIL}, \text{LF}, \text{LDF1}, \text{LDF2})$:

TAIL(j) = 'L'
The lower tail probability is returned, i.e., $p_i = P(F_i \leq f_i : u_i, v_i)$.

TAIL(j) = 'U'
The upper tail probability is returned, i.e., $p_i = P(F_i \geq f_i : u_i, v_i)$.

Constraint: TAIL(j) = 'L' or 'U', for $j = 1, 2, \ldots, \text{LTAIL}$.

3: LF – INTEGER *Input*

On entry: the length of the array F.

Constraint: LF > 0.

4: F(LF) – REAL (KIND=nag_wp) array *Input*

On entry: f_i, the value of the F variate with $f_i = \text{F}(j)$, $j = ((i-1) \bmod \text{LF}) + 1$.

Constraint: F$(j) \geq 0.0$, for $j = 1, 2, \ldots, \text{LF}$.

5: LDF1 – INTEGER *Input*

On entry: the length of the array DF1.

Constraint: LDF1 > 0.

6: DF1(LDF1) – REAL (KIND=nag_wp) array *Input*

On entry: u_i, the degrees of freedom of the numerator variance with $u_i = \text{DF1}(j)$, $j = ((i-1) \bmod \text{LDF1}) + 1$.

Constraint: DF1$(j) > 0.0$, for $j = 1, 2, \ldots, \text{LDF1}$.

7: LDF2 – INTEGER *Input*

On entry: the length of the array DF2.

Constraint: LDF2 > 0.

8: DF2(LDF2) – REAL (KIND=nag_wp) array *Input*

On entry: v_i, the degrees of freedom of the denominator variance with $v_i = \text{DF2}(j)$, $j = ((i-1) \bmod \text{LDF2}) + 1$.

Constraint: DF2$(j) > 0.0$, for $j = 1, 2, \ldots, \text{LDF2}$.

9: P$(*)$ – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array P must be at least max(LTAIL, LF, LDF1, LDF2).

On exit: p_i, the probabilities for the F-distribution.

10: IVALID$(*)$ – INTEGER array *Output*

Note: the dimension of the array IVALID must be at least max(LTAIL, LF, LDF1, LDF2).

On exit: IVALID(i) indicates any errors with the input arguments, with

IVALID$(i) = 0$
No error.

IVALID$(i) = 1$
On entry, invalid value supplied in TAIL when calculating p_i.

IVALID$(i) = 2$

On entry, $f_i < 0.0$.

IVALID$(i) = 3$

On entry, $u_i \leq 0.0$,
or $v_i \leq 0.0$.

IVALID$(i) = 4$

The solution has failed to converge. The result returned should represent an approximation to the solution.

11: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if IFAIL $\neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Note: G01SDF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, at least one value of F, DF1, DF2 or TAIL was invalid, or the solution failed to converge.
Check IVALID for more information.

IFAIL $= 2$

On entry, array size $= \langle value \rangle$.
Constraint: LTAIL > 0.

IFAIL $= 3$

On entry, array size $= \langle value \rangle$.
Constraint: LF > 0.

IFAIL $= 4$

On entry, array size $= \langle value \rangle$.
Constraint: LDF1 > 0.

IFAIL $= 5$

On entry, array size $= \langle value \rangle$.
Constraint: LDF2 > 0.

IFAIL $= -99$

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

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

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

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The result should be accurate to five significant digits.

8 Parallelism and Performance

G01SDF is not threaded in any implementation.

9 Further Comments

For higher accuracy G01SEF can be used along with the transformations given in Section 3.

10 Example

This example reads values from, and degrees of freedom for, a number of F-distributions and computes the associated lower tail probabilities.

10.1 Program Text

```
    Program g01sdfe
!     G01SDF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g01sdf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, ldf1, ldf2, lf, lout,      &
                                          ltail
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: df1(:), df2(:), f(:), p(:)
      Integer, Allocatable             :: ivalid(:)
      Character (1), Allocatable       :: tail(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, mod, repeat
!     .. Executable Statements ..
      Write (nout,*) 'G01SDF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Read in the input vectors
      Read (nin,*) ltail
```

```
      Allocate (tail(ltail))
      Read (nin,*) tail(1:ltail)

      Read (nin,*) lf
      Allocate (f(lf))
      Read (nin,*) f(1:lf)

      Read (nin,*) ldf1
      Allocate (df1(ldf1))
      Read (nin,*) df1(1:ldf1)

      Read (nin,*) ldf2
      Allocate (df2(ldf2))
      Read (nin,*) df2(1:ldf2)

!     Allocate memory for output
      lout = max(lf,ldf1,ldf2,ltail)
      Allocate (p(lout),ivalid(lout))

!     Calculate probability
      ifail = -1
      Call g01sdf(ltail,tail,lf,f,ldf1,df1,ldf2,df2,p,ivalid,ifail)

      If (ifail==0 .Or. ifail==1) Then
!       Display titles
        Write (nout,*)                                                    &
          '   TAIL      F        DF1       DF2       P      IVALID'
        Write (nout,*) repeat('-',56)

!       Display results
        Do i = 1, lout
          Write (nout,99999) tail(mod(i-1,ltail)+1), f(mod(i-1,lf)+1),    &
            df1(mod(i-1,ldf1)+1), df2(mod(i-1,ldf2)+1), p(i), ivalid(i)
        End Do
      End If

99999 Format (5X,A1,3(4X,F6.2),4X,F6.3,4X,I3)
    End Program g01sdfe
```

10.2 Program Data

```
G01SDF Example Program Data
1                       :: LTAIL
'L'                     :: TAIL
3                       :: LF1
5.5 39.9 2.5            :: F
3                       :: LDF1
1.5 1.0 20.25           :: DF1
3                       :: LDF2
25.5 1.0 1.0            :: DF2
```

10.3 Program Results

```
 G01SDF Example Program Results

    TAIL      F        DF1       DF2       P      IVALID
 ------------------------------------------------------------
     L      5.50      1.50     25.50     0.984      0
     L     39.90      1.00      1.00     0.900      0
     L      2.50     20.25      1.00     0.534      0
```