NAG Library Routine Document

G01SFF

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

G01SFF returns a number of lower or upper tail probabilities for the gamma distribution.

2 Specification

```
SUBROUTINE G01SFF (LTAIL, TAIL, LG, G, LA, A, LB, B, P, IVALID, IFAIL)
INTEGER            LTAIL, LG, LA, LB, IVALID(*), IFAIL
REAL (KIND=nag_wp) G(LG), A(LA), B(LB), P(*)
CHARACTER(1)       TAIL(LTAIL)
```

3 Description

The lower tail probability for the gamma distribution with parameters α_i and β_i, $P(G_i \leq g_i)$, is defined by:

$$P\left(G_i \leq g_i : \alpha_i, \beta_i\right) = \frac{1}{{\beta_i}^{\alpha_i}\Gamma(\alpha_i)}\int_0^{g_i} {G_i}^{\alpha_i - 1} e^{-G_i/\beta_i}\, dG_i, \quad \alpha_i > 0.0,\ \beta_i > 0.0.$$

The mean of the distribution is $\alpha_i\beta_i$ and its variance is $\alpha_i{\beta_i}^2$. The transformation $Z_i = \dfrac{G_i}{\beta_i}$ is applied to yield the following incomplete gamma function in normalized form,

$$P\left(G_i \leq g_i : \alpha_i, \beta_i\right) = P\left(Z_i \leq g_i/\beta_i : \alpha_i, 1.0\right) = \frac{1}{\Gamma(\alpha_i)}\int_0^{g_i/\beta_i} {Z_i}^{\alpha_i - 1} e^{-Z_i}\, dZ_i.$$

This is then evaluated using S14BAF.

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

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 a lower or upper tail probability is required. For $j = ((i-1) \bmod \text{LTAIL}) + 1$, for $i = 1, 2, \ldots, \max(\text{LTAIL}, \text{LG}, \text{LA}, \text{LB})$:

TAIL(j) = 'L'
The lower tail probability is returned, i.e., $p_i = P\left(G_i \leq g_i : \alpha_i, \beta_i\right)$.

TAIL(j) = 'U'
The upper tail probability is returned, i.e., $p_i = P(G_i \geq g_i : \alpha_i, \beta_i)$.

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

3: LG – INTEGER *Input*

On entry: the length of the array G.

Constraint: LG > 0.

4: G(LG) – REAL (KIND=nag_wp) array *Input*

On entry: g_i, the value of the gamma variate with $g_i =$ G(j), $j = ((i-1) \bmod \text{LG}) + 1$.

Constraint: G(j) ≥ 0.0, for $j = 1, 2, \ldots,$ LG.

5: LA – INTEGER *Input*

On entry: the length of the array A.

Constraint: LA > 0.

6: A(LA) – REAL (KIND=nag_wp) array *Input*

On entry: the parameter α_i of the gamma distribution with $\alpha_i =$ A(j), $j = ((i-1) \bmod \text{LA}) + 1$.

Constraint: A(j) > 0.0, for $j = 1, 2, \ldots,$ LA.

7: LB – INTEGER *Input*

On entry: the length of the array B.

Constraint: LB > 0.

8: B(LB) – REAL (KIND=nag_wp) array *Input*

On entry: the parameter β_i of the gamma distribution with $\beta_i =$ B(j), $j = ((i-1) \bmod \text{LB}) + 1$.

Constraint: B(j) > 0.0, for $j = 1, 2, \ldots,$ LB.

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

Note: the dimension of the array P must be at least max(LG, LA, LB, LTAIL).

On exit: p_i, the probabilities of the beta distribution.

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

Note: the dimension of the array IVALID must be at least max(LG, LA, LB, LTAIL).

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, $g_i < 0.0$.

IVALID(i) = 3

On entry, $\alpha_i \leq 0.0$,
or $\beta_i \leq 0.0$.

IVALID$(i) = 4$
The solution did not converge in 600 iterations, see S14BAF for details. The probability returned should be a reasonable 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, if you are not familiar with this argument, the recommended value is 0. **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).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, at least one value of G, A, B or TAIL was invalid, or the solution did not 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: LG > 0.

IFAIL $= 4$

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

IFAIL $= 5$

On entry, array size $= \langle value \rangle$.
Constraint: LB > 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 have a relative accuracy of ***machine precision***. There are rare occasions when the relative accuracy attained is somewhat less than ***machine precision*** but the error should not exceed more than 1 or 2 decimal places.

8 Parallelism and Performance

G01SFF is not threaded in any implementation.

9 Further Comments

The time taken by G01SFF to calculate each probability varies slightly with the input arguments g_i, α_i and β_i.

10 Example

This example reads in values from a number of gamma distributions and computes the associated lower tail probabilities.

10.1 Program Text

```
    Program g01sffe
!     G01SFF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g01sff, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, la, lb, lg, lout, ltail
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), b(:), g(:), p(:)
      Integer, Allocatable             :: ivalid(:)
      Character (1), Allocatable       :: tail(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, mod, repeat
!     .. Executable Statements ..
      Write (nout,*) 'G01SFF 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,*) lg
      Allocate (g(lg))
      Read (nin,*) g(1:lg)

      Read (nin,*) la
      Allocate (a(la))
      Read (nin,*) a(1:la)
```

```
      Read (nin,*) lb
      Allocate (b(lb))
      Read (nin,*) b(1:lb)

!     Allocate memory for output
      lout = max(ltail,lg,la,lb)
      Allocate (p(lout),ivalid(lout))

!     Calculate probability
      ifail = -1
      Call g01sff(ltail,tail,lg,g,la,a,lb,b,p,ivalid,ifail)

      If (ifail==0 .Or. ifail==1) Then
!       Display titles
        Write (nout,*)                                                    &
          '  TAIL      G         A         B         P     IVALID'
        Write (nout,*) repeat('-',57)

!       Display results
        Do i = 1, lout
          Write (nout,99999) tail(mod(i-1,ltail)+1), g(mod(i-1,lg)+1),    &
            a(mod(i-1,la)+1), b(mod(i-1,lb)+1), p(i), ivalid(i)
        End Do
      End If

99999 Format (5X,A1,3(4X,F6.2),4X,F6.3,4X,I3)
    End Program g01sffe
```

10.2 Program Data

```
G01SFF Example Program Data
1                       :: LTAIL
'L'                     :: TAIL
4                       :: LG
15.5 0.5 10.0 5.0       :: G
4                       :: LA
 4.0 4.0 1.0 2.0        :: A
4                       :: LB
2.0 1.0 2.0 2.0         :: B
```

10.3 Program Results

```
 G01SFF Example Program Results

   TAIL      G         A         B         P     IVALID
 -------------------------------------------------------
      L     15.50      4.00      2.00     0.950      0
      L      0.50      4.00      1.00     0.002      0
      L     10.00      1.00      2.00     0.993      0
      L      5.00      2.00      2.00     0.713      0
```