NAG Library Routine Document

G01TEF

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

G01TEF returns a number of deviates associated with given probabilities of the beta distribution.

2 Specification

```
SUBROUTINE G01TEF (LTAIL, TAIL, LP, P, LA, A, LB, B, TOL, BETA, IVALID,         &
                   IFAIL)
INTEGER            LTAIL, LP, LA, LB, IVALID(*), IFAIL
REAL (KIND=nag_wp) P(LP), A(LA), B(LB), TOL, BETA(*)
CHARACTER(1)       TAIL(LTAIL)
```

3 Description

The deviate, β_{p_i}, associated with the lower tail probability, p_i, of the beta distribution with parameters a_i and b_i is defined as the solution to

$$P\left(B_i \leq \beta_{p_i} : a_i, b_i\right) = p_i = \frac{\Gamma(a_i + b_i)}{\Gamma(a_i)\Gamma(b_i)} \int_0^{\beta_{p_i}} {B_i}^{a_i - 1}(1 - B_i)^{b_i - 1}\, dB_i, \qquad 0 \leq \beta_{p_i} \leq 1;\ a_i, b_i > 0.$$

The algorithm is a modified version of the Newton–Raphson method, following closely that of Cran *et al.* (1977).

An initial approximation, β_{i0}, to β_{p_i} is found (see Cran *et al.* (1977)), and the Newton–Raphson iteration

$$\beta_k = \beta_{k-1} - \frac{f_i(\beta_{k-1})}{f_i'(\beta_{k-1})},$$

where $f_i(\beta_k) = P\left(B_i \leq \beta_k : a_i, b_i\right) - p_i$ is used, with modifications to ensure that β_k remains in the range $(0, 1)$.

The input arrays to this routine are designed to allow maximum flexibility in the supply of vector parameters 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

Cran G W, Martin K J and Thomas G E (1977) Algorithm AS 109. Inverse of the incomplete beta function ratio *Appl. Statist.* **26** 111–114

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

5 Parameters

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 which tail the supplied probabilities represent. For $j = ((i-1) \bmod \text{LTAIL}) + 1$, for $i = 1, 2, \ldots, \max(\text{LTAIL}, \text{LP}, \text{LA}, \text{LB})$:

TAIL(j) = 'L'
The lower tail probability, i.e., $p_i = P\left(B_i \leq \beta_{p_i} : a_i, b_i\right)$.

TAIL(j) = 'U'
The upper tail probability, i.e., $p_i = P\left(B_i \geq \beta_{p_i} : a_i, b_i\right)$.

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

3: LP – INTEGER *Input*

On entry: the length of the array P.

Constraint: LP > 0.

4: P(LP) – REAL (KIND=nag_wp) array *Input*

On entry: p_i, the probability of the required beta distribution as defined by TAIL with $p_i = \text{P}(j)$, $j = ((i-1) \bmod \text{LP}) + 1$.

Constraint: $0.0 \leq \text{P}(j) \leq 1.0$, for $j = 1, 2, \ldots, \text{LP}$.

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: a_i, the first parameter of the required beta distribution with $a_i = \text{A}(j)$, $j = ((i-1) \bmod \text{LA}) + 1$.

Constraint: $0.0 < \text{A}(j) \leq 10^6$, for $j = 1, 2, \ldots, \text{LA}$.

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: b_i, the second parameter of the required beta distribution with $b_i = \text{B}(j)$, $j = ((i-1) \bmod \text{LB}) + 1$.

Constraint: $0.0 < \text{B}(j) \leq 10^6$, for $j = 1, 2, \ldots, \text{LB}$.

9: TOL – REAL (KIND=nag_wp) *Input*

On entry: the relative accuracy required by you in the results. If G01TEF is entered with TOL greater than or equal to 1.0 or less than $10 \times$ ***machine precision*** (see X02AJF), then the value of $10 \times$ ***machine precision*** is used instead.

10: BETA($*$) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array BETA must be at least $\max(\text{LTAIL}, \text{LP}, \text{LA}, \text{LB})$.

On exit: β_{p_i}, the deviates for the beta distribution.

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

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

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 β_{p_i}.

IVALID(i) $= 2$

On entry, $p_i < 0.0$,
or $p_i > 1.0$.

IVALID(i) $= 3$

On entry, $a_i \leq 0.0$,
or $a_i > 10^6$,
or $b_i \leq 0.0$,
or $b_i > 10^6$.

IVALID(i) $= 4$
The solution has not converged but the result should be a reasonable approximation to the solution.

IVALID(i) $= 5$
Requested accuracy not achieved when calculating the beta probability. The result should be a reasonable approximation to the correct solution.

12: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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 parameters 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: G01TEF 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 TAIL, P, A, or B 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 = ⟨*value*⟩.
Constraint: LP > 0.

IFAIL = 4

On entry, array size = ⟨*value*⟩.
Constraint: LA > 0.

IFAIL = 5

On entry, array size = ⟨*value*⟩.
Constraint: LB > 0.

IFAIL = −999

Dynamic memory allocation failed.

7 Accuracy

The required precision, given by TOL, should be achieved in most circumstances.

8 Further Comments

The typical timing will be several times that of G01SEF and will be very dependent on the input parameter values. See G01SEF for further comments on timings.

9 Example

This example reads lower tail probabilities for several beta distributions and calculates and prints the corresponding deviates.

9.1 Program Text

```
    Program g01tefe
!     G01TEF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: g01tef, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: tol
      Integer                          :: i, ifail, la, lb, lout, lp, ltail
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:), b(:), beta(:), p(:)
      Integer, Allocatable             :: ivalid(:)
      Character (1), Allocatable       :: tail(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, mod, repeat
!     .. Executable Statements ..
      Write (nout,*) 'G01TEF Example Program Results'
      Write (nout,*)

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

!     Read in the tolerance
      Read (nin,*) tol
```

```
!     Read in the input vectors
      Read (nin,*) ltail
      Allocate (tail(ltail))
      Read (nin,*) tail(1:ltail)

      Read (nin,*) lp
      Allocate (p(lp))
      Read (nin,*) p(1:lp)

      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,la,lb,lp)
      Allocate (beta(lout),ivalid(lout))

!     Calculate deviates (inverse CDF)
      ifail = -1
      Call g01tef(ltail,tail,lp,p,la,a,lb,b,tol,beta,ivalid,ifail)

      If (ifail==0 .Or. ifail==1) Then
!       Display titles
        Write (nout,*) &
          '   TAIL      P         A         B         BETA   IVALID'
        Write (nout,*) repeat('-',55)

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

99999 Format (5X,A1,4X,F6.3,2(4X,F6.2),3X,F7.3,4X,I3)
    End Program g01tefe
```

9.2 Program Data

```
G01TEF Example Program Data
0.0                     :: TOL
1                       :: LTAIL
'L'                     :: TAIL
3                       :: LP
0.5 0.99 0.25           :: P
3                       :: LA
1.0 1.5 20.0            :: A
3                       :: LB
2.0 1.5 10.0            :: B
```

9.3 Program Results

```
 G01TEF Example Program Results

    TAIL      P         A         B         BETA   IVALID
 ------------------------------------------------------------
     L     0.500      1.00      2.00     0.293      0
     L     0.990      1.50      1.50     0.967      0
     L     0.250     20.00     10.00     0.611      0
```