National Bureau of Standards
Statistical Engineering Division
Gaithersburg, MD 20899

N-86-5
December 1986

An Algorithm for Computing the Doubly Noncentral t C.D.F. to a Specified Accuracy

Charles P. Reeve

Let Z be a normally distributed random variable with mean δ and variance 1, and X be a noncentral chi-squared random variable, independent of Z, with degrees of freedom $\nu>0$ and noncentrality parameter $\lambda\geq 0$. Then the random variable

$$Y = Z/\sqrt{X/\nu} \tag{1}$$

has the doubly noncentral t distribution, indicated by $Y \sim t''(\nu,\delta,\lambda)$. This distribution was introduced by Robbins [14] as the distribution of Student's t statistic when the observations have unequal population means. It was later used by Patnaik [11] in testing hypotheses concerning the standardized means of nonhomogeneous normal populations.

Krishnan [7] gives a series representation for the t" cumulative distribution function (c.d.f.) in terms of incomplete beta functions. Alternative series representations are given by Bulgren and Amos [2], Bulgren [3], Carey [4], and [7]. Approximations are given by Johnson and Kotz [6] and Mulholkar and Chaubey [8]. Numerical examples of usage are given in [3] and [7].

The author has been unable to find any published algorithms for computing exact values of the t" c.d.f. although computer programs have obviously been used in generating published tables in [2,3,4,7,8]. The purpose of this note is to present an efficient algorithm for computing the t" c.d.f. to a specified accuracy using exact formulas.

The algorithm uses the series representation in eq. (4) of [7] which can be re-written

$$F_Y(x) = \sum_{j=0}^{\infty} \sum_{i=0}^{\infty} A_j B_i I(u,1/2+i/2,\nu/2+j)/2 + \sum_{j=0}^{\infty} A_j \sum_{i=0}^{\infty} B_i(-1)^i/2 \tag{2}$$

where $A_j = (\lambda/2)^j e^{-\lambda/2}/\Gamma(j+1)$, $B_i = (\delta/\sqrt{2})^i e^{-\delta^2/2}/\Gamma(i/2+1)$, $u = x^2/(x^2+\nu)$, and $x\geq 0$. When $x<0$ the c.d.f. is computed from the relation $F_Y(x;\nu,\delta,\lambda) = 1 - F_Y(-x;\nu,-\delta,\lambda)$. The A_j are Poisson probabilities, and

$I(u,a,b) = \int_0^u t^{a-1}(1-t)^{b-1}dt/B(a,b)$ is the c.d.f. of the beta distribution (also called the incomplete beta ratio) where $0<u<1$, $a>0$, $b>0$, and

$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}dt$. The quantity $(\delta/\sqrt{2})^i$ in B_i is erroneously given as $(\delta^2/2)^{i/2}$ in [7]. In the latter form the quantity would be (incorrectly) positive when δ is negative and i is an odd integer.

Each summation over i in (2) can be split into two summations over even and odd values of i. For i=0,1,2,... let $B_i^e = B_{2i} = (\delta^2/2)^i e^{-\delta^2/2}/\Gamma(i+1)$ and $B_i^o = B_{2i+1} = (\delta/\sqrt{2})(\delta^2/2)^i e^{-\delta^2/2}/\Gamma(i+3/2)$. Then (2) takes the form

$$F_Y(x) = \sum_{j=0}^{\infty} \sum_{i=0}^{\infty} A_j B_i^e I(u,1/2+i,\nu/2+j)/2$$
$$+ \sum_{j=0}^{\infty} \sum_{i=0}^{\infty} A_j B_i^o I(u,1+i,\nu/2+j)/2 + \{1 - \sum_{i=0}^{\infty} B_i^o\}/2 \qquad (3)$$

since $\sum_{j=0}^{\infty} A_j = \sum_{i=0}^{\infty} B_i^e = 1$. For computational purposes the infinite series must be truncated, thus (3) is re-expressed as

$$F_Y(x) = \sum_{j=J'}^{J''} \sum_{i=I_e'}^{I''} A_j B_i^e I(u,1/2+i,\nu/2+j)/2$$
$$+ \sum_{j=J'}^{J''} \sum_{i=I_o'}^{I''} A_j B_i^o I(u,1+i,\nu/2+j)/2 + \{1 - \sum_{i=I_o'}^{I''} B_i^o\}/2 + R \qquad (4)$$

where I_e', I_o', I'', J', and J'' are non-negative integers and R is the remainder. If the beta c.d.f.'s are computed without error, it can easily be shown that choosing I_e', I_o', I'', J', and J'' such that $\sum_{j=J'}^{J''} A_j \geq 1-2\varepsilon/3$, $\sum_{i=I_e'}^{I''} B_i^e \geq 1-2\varepsilon/3$, and $I_o' = \max\{I_e'-1,0\}$ yields $R<\varepsilon$ provided $\varepsilon>0$. Therefore, ε serves as an absolute error bound on $F_Y(x)$.

For maximum computational efficiency, the number of terms in each sum is minimized by indexing j and i over the largest of the Poisson probabilities A_j and B_i^e respectively. It then follows that i also indexes over the B_i^o which are largest in absolute value.

The final task is to compute the $(2I''-I_e'-I_o'+2)(J''-J'+1)$ beta c.d.f.'s and the summations. An efficient procedure for doing this is to first compute only $I(u,1/2+I_e',\nu/2+J_e^*)$ and $I(u,1/2+I_e^*,\nu/2+J')$ directly, indicated by the symbols "x" and "y" in figure 1. The remaining beta c.d.f's are computed using the recurrence relations

$$I(x,a,b) = I(x,a,b+1) - x^a(1-x)^b/[bB(a,b)], \tag{5a}$$

$$I(x,a,b) = I(x,a+1,b) + x^a(1-x)^b/[aB(a,b)], \text{ and} \tag{5b}$$

$$I(x,a,b) = xI(x,a-1,b) + (1-x)I(x,a,b-1) \tag{5c}$$

as found in Abramowitz and Stegun [1]. Subject to the restrictions $J' \leq J_e^* \leq J''$ and $I_e' \leq I_e^* \leq I''$, J_e^* and I_e^* are chosen to maximize the magnitudes of the rightmost terms in (5a) and (5b) respectively. In applying each of these two recurrence relations only one direct evaluation of B(a,b) is necessary, computed by $B(a,b) = e^{\ln\Gamma(a) + \ln\Gamma(b) - \ln\Gamma(a+b)}$ where $\Gamma(\alpha) = \int_0^\infty t^{\alpha-1}e^{-t}dt$. Further values are easily computed from the identities $B(a+1,b) = aB(a,b)/(a+b)$ and $B(a,b+1) = bB(a,b)/(a+b)$. In a similar fashion $I(u,1+I_o',\nu/2+J_o^*)$ and $I(u,1+I_o^*,\nu/2+J')$ are computed directly, and the same procedure is followed with similar restrictions on J_o^* and I_o^*. These computations are illustrated in in figure 2. The double summations in (4) are accumulated as the beta c.d.f.'s are computed.

In figures 1 and 2 the symbols "a", "b", and "c" indicate which of the recurrence relations (4) is used in computing each beta c.d.f., with those indicated by "c" being done last.

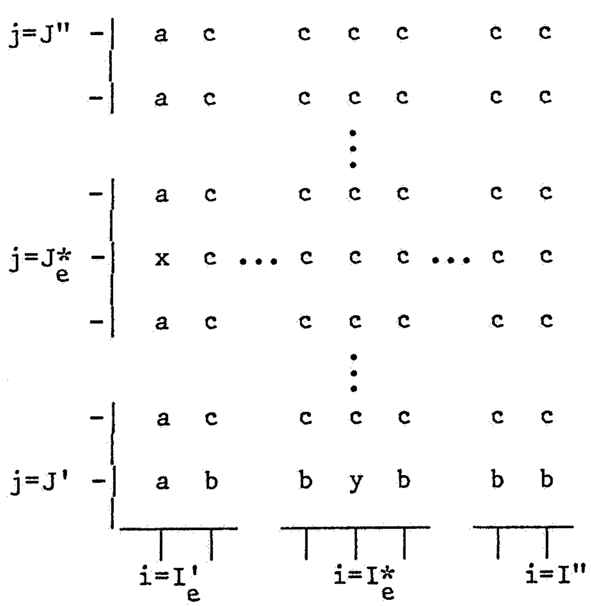

Figure 1

Computation of the I(u,1/2+i,ν/2+j)

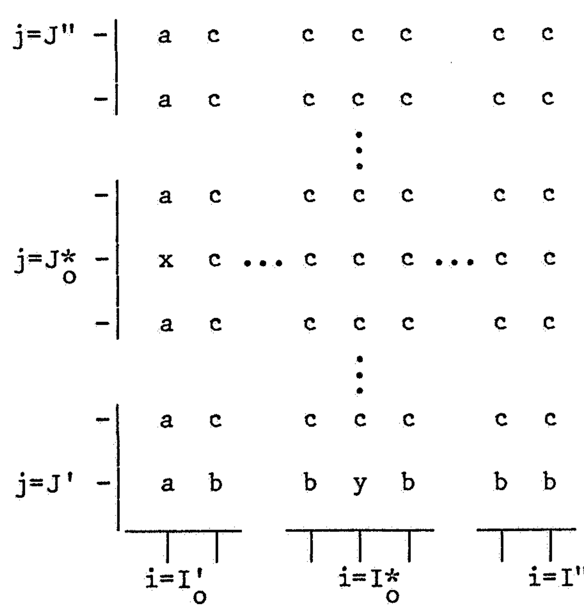

Figure 2

Computation of the I(u,1+i,ν/2+j)

The above algorithm has been incorporated into the FORTRAN subroutine CDFDNT. External routines for computing the beta c.d.f. and the double precision log of the gamma function are required. In the current version of CDFNDT these routines are the subroutine CDFBET and the function GAMLOG as described in Reeve [12,13]. Other routines which can be substituted for CDFBET and GAMLOG are those in [5,9,10]. For all practical purposes the absolute error criterion ε will be met if the beta c.d.f. routine is accurate to two or three digits beyond ε.

The recursive method of computing the beta c.d.f.'s requires a little extra computer programming and storage, but results in a tremendous savings in computing time as δ and λ become large. The computing time and storage also increase as ε becomes small, but are unaffected by ν. In table 1 the t" c.d.f. is computed for selected parameter values. The computing time in CPU seconds and the number of beta c.d.f.'s computed are included. The case $\nu=\delta=\lambda=100$ corresponds to the example in Carey [4] who defines λ a bit differently. Her single series representation of the t" c.d.f. appears well suited for computation when δ and/or λ take on large values. Note that CDFDNT required only 0.76 CPU seconds in this case. Were all 98,490 beta c.d.f. evaluations done by separate calls to the beta c.d.f. routine, the CPU time would have been at least 100 times greater. The computations in table 1 were done on the CDC Cyber 180/855 computer at NBS.

The current dimension limits in CDFDNT allow values of δ up to 100 and λ up to 10,000 with ε as small as 10^{-8}, but these limits could easily be increased by the user. The limiting factor in using CDFDNT is more likely to be execution time than storage.

Steps were taken to eliminate underflow situations, minimize the effects of roundoff error, and minimize storage requirements. Only those c.d.f. values indicated by "x", "y", "a", or "b" in figures 1 and 2 are actually stored. The Poisson probabilities $A_{J'}, \ldots, A_{J''}$ and $B^e_{I'_e}, \ldots, B^e_{I''}$ are also stored as are $B^o_{I'_o}, \ldots, B^o_{I''}$.

If $\lambda=0$ then the doubly noncentral t reduces to the (singly) noncentral t, and if $\delta=\lambda=0$ it reduces to the central t. In either case, CDFDNT will run almost as efficiently as routines designed for those specific cases.

Portions of tables in [2,3,4,7,8] were reproduced by CDFDNT and agreed to within roundoff error in each case.

A listing of CDFDNT is an appendix to this note. It is invoked by

```
CALL CDFDNT(X,DF,DELTA,ALAMB,EPS,IFLAG,CDFX)
```

where the arguments are defined in the program documentation. The returned value of CDFX is valid only if IFLAG=0 on return. In passing ε (variable name EPS) to CDFDNT the user should realize that accuracy is limited by the number of digits carried in a single precision variable, and that roundoff error may affect the last one or two of these digits.

Table 1

Computing times on the CDC Cyber 180/855 for the c.d.f. of $t''(\nu,\delta,\lambda)$ using CDFDNT for selected parameter values.

$\varepsilon=10^{-6}$

ν	δ	λ	x †	$P\{t''\leq x\}$	CPU sec	No. beta c.d.f. values
1	1	1	0.7071	0.433771	0.01	128
1	1	100	0.0995	0.498015	0.02	1,120
1	1	10000	0.0100	0.500000	0.10	11,248
1	10	1	7.0711	0.349271	0.02	1,128
1	10	100	0.9950	0.485863	0.08	9,870
1	10	10000	0.1000	0.500000	0.67	99,123
1	100	1	70.7107	0.347264	0.11	11,256
1	100	100	9.9504	0.480221	0.66	98,490
1	100	10000	1.0000	0.500000	6.47	989,121
10	1	1	0.9535	0.490326	0.01	128
10	1	100	0.3015	0.498251	0.01	1,120
10	1	10000	0.0316	0.499892	0.10	11,248
10	10	1	9.5346	0.448390	0.02	1,128
10	10	100	3.0151	0.487089	0.08	9,870
10	10	10000	0.3161	0.500181	0.75	99,123
10	100	1	95.3463	0.441153	0.12	11,256
10	100	100	30.1511	0.480930	0.65	98,490
10	100	10000	3.1607	0.498611	6.48	989,121
100	1	1	0.9950	0.498990	0.01	128
100	1	100	0.7071	0.499248	0.01	1,120
100	1	10000	0.0995	0.499965	0.12	11,248
100	10	1	9.9504	0.490966	0.02	1,128
100	10	100	7.0711	0.493307	0.08	9,870
100	10	10000	0.9950	0.499656	0.73	99,123
100	100	1	99.5037	0.481469	0.13	11,256
‡ 100	100	100	70.7107	0.485762	0.76	98,490
100	100	10000	9.9504	0.498682	6.71	989,121

† $x = \delta/\sqrt{1 + \lambda/\nu}$ rounded to four decimal places

‡ example in Carey [4] with large values of ν, δ, and λ

References

1. Abramowitz, Milton and Stegun, Irene A., Handbook of Mathematical Functions, NBS Special Publication 55, 1970, p. 944.

2. Bulgren, W.G. and Amos, D.E., "A Note on Representations of the Doubly Non-Central t Distribution", Journal of the American Statistical Association, Vol. 63, No. 323, September 1968, pp. 1013-1019.

3. Bulgren, W.G., "Probability Integral of the Doubly Noncentral t-Distribution with Degrees of Freedom n and Non-Centrality Parameters δ and λ", Selected Tables in Mathematical Statistics, Vol. II, 1974, pp. 1-138.

4. Carey, Michele B., "Evaluation of the Doubly Noncentral t Cumulative Distribution Function", Computer Science and Statistics - The Interface, James E. Gentle (ed.), North Holland Publishing Company, 1983, pp. 339-343.

5. IMSL, Inc., Houston, TX. [MDBETA, DLGAMA]

6. Johnson, Norman L. and Kotz, Samuel, Continuous Univariate Distributions-2, Houghton Mifflin Company, 1970, pp. 213-215.

7. Krishnan, Marakatha, "Series Representations of the Doubly Noncentral t-Distribution", Journal of the American Statistical Association, Vol. 63, No. 323, September 1968, pp. 1004-1012.

8. Mudholkar, Govind S. and Chaubey, Yogendra P., "A Simple Approximation for the Doubly Noncentral t-Distribution", Communications in Statistics - Simulation and Computation, Vol. B5, Nos. 2&3, 1976, pp. 85-92.

9. NBS Core Math Library (CMLIB). [BETAI, DLNGAM]

10. Numerical Algorithms Group (NAG), Downers Grove, IL. [G01BDE, S14ABF]

11. Patnaik, P.B., "Hypotheses Concerning the Means of Observations in Normal Samples", Sankhya, Vol. 15, 1955, pp. 343-372.

12. Reeve, Charles P., "An Algorithm for Computing the Beta C.D.F. to a Specified Accuracy", SED Note 86-3, October 1986.

13. Reeve, Charles P., "Accurate Computation of the Log of the Gamma Function", SED Note 86-1, October 1986.

14. Robbins, Herbert, "The Distribution of Student's t When the Population Means are Unequal", Annals of Mathematical Statistics, Vol. 19, No. 3, September 1948, pp. 406-410.

```
*CDFDNT
      SUBROUTINE CDFDNT (X,DF,DELTA,ALAMB,EPS,IFLAG,CDFX)
C
C-----------------------------------------------------------------------
C   CDFDNT   WRITTEN BY CHARLES P. REEVE, STATISTICAL ENGINEERING
C            DIVISION, NATIONAL BUREAU OF STANDARDS, GAITHERSBURG,
C            MARYLAND  20899
C
C   FOR: COMPUTING THE CUMULATIVE DISTRIBUTION FUNCTION OF THE DOUBLY
C        NONCENTRAL T DISTRIBUTION TO A SPECIFIED ACCURACY (TRUNCATION
C        ERROR IN THE INFINITE SERIES REPRESENTATION GIVEN BY EQUATION
C        4 IN REFERENCE 1 BELOW).  WHEN X<0 THE C.D.F. IS COMPUTED
C        FROM CDF(X,DF,DELTA,ALAMB) = 1 - CDF(-X,DF,-DELTA,ALAMB).
C        THE BETA C.D.F. ROUTINE IS CALLED AT MOST FOUR TIMES.  FURTHER
C        VALUES OF THE BETA C.D.F. ARE OBTAINED FROM RECURRENCE
C        RELATIONS GIVEN IN REFERENCE 2.  REFERENCE 3 GIVES A DETAILED
C        DESCRIPTION OF THE ALGORITHM HEREIN.
C
C        THIS PROGRAM MAY ALSO BE EFFICIENTLY USED TO COMPUTE THE
C        CUMULATIVE DISTRIBUTION FUNCTIONS OF THE SINGLY NONCENTRAL
C        AND CENTRAL T DISTRIBUTIONS BY SETTING THE APPROPRIATE
C        NONCENTRALITY PARAMETERS EQUAL TO ZERO.
C
C        CHECKS ARE MADE TO ASSURE THAT ALL PASSED PARAMETERS ARE
C        WITHIN VALID RANGES AS GIVEN BELOW.  NO UPPER LIMIT IS SET
C        FOR THE NONCENTRALITY PARAMETERS, BUT VALUES UP TO ABOUT 100
C        FOR DELTA AND 10,000 FOR LAMBDA CAN BE HANDLED WITH THE
C        CURRENT DIMENSION LIMITS.  THE COMPUTED VALUE CDFX IS VALID
C        ONLY IF IFLAG=0 ON RETURN.
C
C   NOTE: IN SUBROUTINE EDGET THE DOUBLE PRECISION CONSTANT DEUFLO IS
C         THE EXPONENTIAL UNDERFLOW LIMIT WHOSE CURRENT VALUE IS SET
C         AT -69D0.  ON A COMPUTER WHERE DEXP(-69D0) CAUSES UNDERFLOW
C         THIS LIMIT SHOULD BE CHANGED.
C
C   SUBPROGRAMS CALLED: CDFBET (BETA C.D.F.)
C                       GAMLOG (DOUBLE PRECISION LOG OF GAMMA FUNCTION)
C                       POISST, EDGET, GRID (ATTACHED)
C
C   CURRENT VERSION COMPLETED SEPTEMBER 29, 1988
C
C   REFERENCES:
C
C   1. KRISHNAN, MARAKATHA, 'SERIES REPRESENTATIONS OF THE DOUBLY
C      NONCENTRAL T DISTRIBUTION', JOURNAL OF THE AMERICAN STATISTICAL
C      ASSOCIATION, SEPTEMBER 1968, VOLUME 63, NO. 323, PP. 1004-1012.
C
C   2. ABRAMOWITZ, MILTON, AND STEGUN, IRENE A., 'HANDBOOK OF
C      MATHEMATICAL FUNCTIONS', NATIONAL BUREAU OF STANDARDS APPLIED
C      MATHEMATICS SERIES 55, NOVEMBER 1970, P. 944.
C
C   3. REEVE, CHARLES P., 'AN ALGORITHM FOR COMPUTING THE DOUBLY
C      NONCENTRAL T C.D.F. TO A SPECIFIED ACCURACY', STATISTICAL
C      ENGINEERING DIVISION NOTE 86-5, DECEMBER 1986.
C-----------------------------------------------------------------------
C   DEFINITION OF PASSED PARAMETERS:
C
C        * X = VALUE AT WHICH THE C.D.F. IS TO BE COMPUTED (REAL)
C
C       * DF = DEGREES OF FREEDOM (>0) IN THE DENOMINATOR (REAL)
C
C    * DELTA = THE NONCENTRALITY PARAMETER FOR THE NUMERATOR (REAL)
C              [EQUAL TO ZERO FOR THE CENTRAL T DISTRIBUTION]
C
C    * ALAMB = THE NONCENTRALITY PARAMETER (>=0) FOR THE DENOMINATOR
C              (REAL) [EQUAL TO ZERO FOR THE SINGLY NONCENTRAL T AND
C              CENTRAL T DISTRIBUTIONS]
C
C      * EPS = THE DESIRED ABSOLUTE ACCURACY OF THE C.D.F. (REAL)
C              [1 >= EPS >= 10**(-10)]
C
C     IFLAG = ERROR INDICATOR ON OUTPUT (INTEGER)   INTERPRETATION:
C                0 -> NO ERRORS DETECTED
C              1,2 -> ERROR FLAGS FROM SUBROUTINE CDFBET
C                3 -> ALAMB IS < 0
C                4 -> DF IS <= 0
C                5 -> EPS IS OUTSIDE THE RANGE [10**(-10),1]
C                6 -> VECTOR DIMENSIONS ARE TOO SMALL - INCREASE NX
C
C      CDFX = THE DOUBLY NONCENTRAL T C.D.F. EVALUATED AT X (REAL)
```

```
C
C   * INDICATES PARAMETERS REQUIRING INPUT VALUES
C-----------------------------------------------------------------------
C
      PARAMETER (NX=1000)
      DIMENSION BFI(NX),BFJ(NX),POI(NX),POJ(NX)
      DOUBLE PRECISION DARG,DFA
      LOGICAL LL
      CDFX = 0.0
C
C---- CHECK VALIDITY OF ARGUMENTS
C
      IF (ALAMB.LT.0.0) THEN
         IFLAG = 3
         RETURN
      ENDIF
      IF (DF.LE.0.0) THEN
         IFLAG = 4
         RETURN
      ENDIF
      IF (EPS.GT.1.0.OR.EPS.LT.1.0E-10) THEN
         IFLAG = 5
         RETURN
      ENDIF
      IFLAG = 0
C
C---- SET ERROR CRITERION FOR THE BETA C.D.F. (PECULIAR TO CDFBET)
C
      EPS3 = 0.001*EPS
C
      DELSQ = DELTA**2
      FA = 0.5*DELSQ
      GA = 0.5*ALAMB
      GB = 0.5*DF
      YY = DF/(DF+X*X)
      XX = 1.0-YY
C
C---- IF X<0 SET LL=.TRUE., REVERSE SIGN OF DELTA, AND USE THE
C---- IDENTITY DESCRIBED UP FRONT FOR COMPUTING THE C.D.F.
C
      LL = X.LT.0.0
      IF (XX.GE.1.0) THEN
         CDFX = 1.0
         GO TO 50
      ENDIF
      SDELTA = DELTA
      IF (LL) SDELTA = -DELTA
C
C---- COMPUTE POISSON PROBABILITIES IN VECTOR POI
C
      CALL POISST (FA,EPS,IMIN,NI,POI,NX,IFLAG)
      IF (IFLAG.NE.0) RETURN
      IF (YY.GE.1.0) GO TO 10
      FC = 0.5+REAL(IMIN)
C
C---- COMPUTE POISSON PROBABILITIES IN VECTOR POJ
C
      CALL POISST (GA,EPS,JMIN,NJ,POJ,NX,IFLAG)
      IF (IFLAG.NE.0) RETURN
      GC = GB+REAL(JMIN)
C
C---- SUM THE TERMS CORRESPONDING TO 'EVEN' VALUES OF INDEX I
C
      CALL GRID (NI,NJ,FC,GC,BFI,BFJ,POI,POJ,XX,YY,EPS3,CDFX,IFLAG)
      IF (IFLAG.NE.0) RETURN
   10 IF (DELTA.EQ.0.0) THEN
         NI = 0
         SUM = 0.0
         IF (YY.GE.1.0) GO TO 40
      ELSE
C
C---- COMPUTE 'POISSON-LIKE' PROBABILITIES IN VECTOR POI
C
         K = INT(FA)
         IF (IMIN.GT.0) THEN
            IMIN = IMIN-1
            NI = NI+1
         ENDIF
         DFA = DBLE(FA)
         DARG = (DBLE(K)+0.5D0)*DLOG(DFA)-DFA-GAMLOG(REAL(K)+1.5)
```

965

K=200
DFA=200

```
         L = K-IMIN+1
         POI(L) = SIGN(SNGL(DEXP(DARG)),SDELTA)
         SUM = POI(L)
         DO 20 I = K-1, IMIN, -1
            L = L-1
            POI(L) = POI(L+1)*(REAL(I)+1.5)/FA
            SUM = SUM+POI(L)
   20    CONTINUE
         L = K-IMIN+1
         DO 30 I = K+1, IMIN+NI-1
            L = L+1
            POI(L) = POI(L-1)*FA/(REAL(I)+0.5)
            SUM = SUM+POI(L)
   30    CONTINUE
         IF (YY.GE.1.0) GO TO 40
         FC = 1.0+REAL(IMIN)
C
C---- SUM THE TERMS CORRESPONDING TO 'ODD' VALUES OF INDEX I
C
         CALL GRID (NI,NJ,FC,GC,BFI,BFJ,POI,POJ,XX,YY,EPS3,CDFX,IFLAG)
         IF (IFLAG.NE.0) RETURN
      ENDIF
C
C---- COMPUTE THE NORMAL C.D.F. AT -SDELTA
C
   40 PHI = 0.5*(1.0-SUM)
C
C---- COMPUTE THE DOUBLY NONCENTRAL T C.D.F. AT X, USING AN IDENTITY
C---- IF X<0
C
      CDFX = 0.5*CDFX+PHI
   50 IF (LL) CDFX = 1.0-CDFX
      RETURN
      END
C
      SUBROUTINE POISST (ALAMB,EPS,L,NSPAN,V,NV,IFLAG)
C
C---- COMPUTE THE POISSON(ALAMB) PROBABILITIES OVER THE RANGE [L,K]
C---- WHERE THE TOTAL TAIL PROBABILITY IS LESS THAN EPS/3, SUM THE
C---- PROBABILITIES IN DOUBLE PRECISION, AND SHIFT THEM TO THE
C---- BEGINNING OF VECTOR V.
C
      DIMENSION V(*)
      DOUBLE PRECISION DAL,DK,DLIMIT,DSUM,GAMLOG
      DLIMIT = 1.0D0-2.0D0*DBLE(EPS)/3.0D0
      K = INT(ALAMB)
      L = K+1
      IF (ALAMB.EQ.0.0) THEN
         PL = 1.0
      ELSE
         DAL = DBLE(ALAMB)
         DK = DBLE(K)
         PL = SNGL(DEXP(DK*DLOG(DAL)-DAL-GAMLOG(REAL(K+1))))
      ENDIF
      PK = ALAMB*PL/REAL(L)
      NK = NV/2
      NL = NK+1
      DSUM = 0.0
   10 IF (PL.LT.PK) THEN
         NK = NK+1
         IF (NK.GT.NV) THEN
            IFLAG = 6
            RETURN
         ENDIF
         V(NK) = PK
         DSUM = DSUM+DBLE(PK)
         K = K+1
         IF (DSUM.GE.DLIMIT) GO TO 20
         PK = ALAMB*PK/REAL(K+1)
      ELSE
         NL = NL-1
         V(NL) = PL
         DSUM = DSUM+DBLE(PL)
         L = L-1
         IF (DSUM.GE.DLIMIT) GO TO 20
         PL = REAL(L)*PL/ALAMB
      ENDIF
      GO TO 10
   20 INC = NL-1
      DO 30 I = NL, NK
```

```
         V(I-INC) = V(I)
   30 CONTINUE
      NSPAN = NK-INC
      RETURN
      END
C
      SUBROUTINE EDGET (NK,FC,GC,XX,YY,BFK,CDFX,POI,POJ,EPS3,IFLAG,L)
C
C---- COMPUTE THE BETA C.D.F.'S BY A RECURRENCE RELATION ALONG THE EDGES
C---- I = IMIN AND J = JMIN OF A GRID.  THE CORRESPONDING COMPONENTS OF
C---- THE T" C.D.F. ARE INCLUDED IN THE SUMMATION.  TERMS WHICH MIGHT
C---- CAUSE UNDERFLOW ARE SET TO ZERO.
C
      DIMENSION BFK(*),POI(*),POJ(*)
      DOUBLE PRECISION DARG,DEUFLO,GAMLOG
      DATA DEUFLO / -69.0D0 /
      FD = FC-1.0
      K = MAX0(L,MIN0(NK,INT((GC-1.0)*XX/YY-FD)))
      FK = FD+REAL(K)
      CALL CDFBET (XX,FK,GC,EPS3,IFLAG,BFK(K))
      IF (IFLAG.NE.0) RETURN
      IF (L.EQ.1) BFK(K) = 1.0-BFK(K)
      IF (NK.EQ.1) GO TO 40
      DARG = DBLE(FK)*DLOG(DBLE(XX))+DBLE(GC)*DLOG(DBLE(YY))-
     *   DLOG(DBLE(FK))+GAMLOG(FK+GC)-GAMLOG(FK)-GAMLOG(GC)
      IF (DARG.LT.DEUFLO) THEN
         DK = 0.0
      ELSE
         DK = SNGL(DEXP(DARG))*(-1.0)**L
      ENDIF
      IF (K.GE.NK) GO TO 20
      BFK(K+1) = BFK(K)-DK
      DI = DK
      KFLAG = 1
      DO 10 I = K+1, NK-1
         IF (KFLAG.EQ.1) THEN
            DI = DI*(FD+GC+REAL(I-1))*XX/(FD+REAL(I))
            IF (DK+DI.EQ.DK) THEN
               KFLAG = 0
               DI = 0.0
            ENDIF
         ENDIF
         BFK(I+1) = BFK(I)-DI
   10 CONTINUE
   20 DI = DK
      KFLAG = 1
      DO 30 I = K-1, L, -1
         IF (KFLAG.EQ.1) THEN
            DI = DI*(FC+REAL(I))/((FD+GC+REAL(I))*XX)
            IF (DK+DI.EQ.DK) THEN
               KFLAG = 0
               DI = 0.0
            ENDIF
         ENDIF
         BFK(I) = BFK(I+1)+DI
   30 CONTINUE
   40 DO 50 I = L, NK
         CDFX = CDFX+POI(I)*POJ(1)*BFK(I)
   50 CONTINUE
      RETURN
      END
C
      SUBROUTINE GRID (NI,NJ,FC,GC,BFI,BFJ,POI,POJ,XX,YY,EPS3,CDFX,IFLAG
     *   )
C
C---- COMPUTE DOUBLE SUMMATION OF COMPONENTS OF THE T" C.D.F. OVER THE
C---- GRID I=IMIN TO IMAX AND J=JMIN TO JMAX
C
      DIMENSION BFI(*),BFJ(*),POI(*),POJ(*)
C
C---- COMPUTE BETA C.D.F. BY RECURRENCE WHEN I=IMIN, J=JMIN TO JMAX
C
      CALL EDGET (NJ,GC,FC,YY,XX,BFJ,CDFX,POJ,POI,EPS3,IFLAG,1)
      IF (NI.LE.1.OR.IFLAG.NE.0) RETURN
C
C---- COMPUTE BETA C.D.F. BY RECURRENCE WHEN J=JMIN, I=IMIN TO IMAX
C
      BFI(1) = BFJ(1)
      CALL EDGET (NI,FC,GC,XX,YY,BFI,CDFX,POI,POJ,EPS3,IFLAG,2)
      IF (NJ.LE.1.OR.IFLAG.NE.0) RETURN
```

```
C
C--- COMPUTE BETA C.D.F. BY RECURRENCE WHEN I>IMIN, J>JMIN
C
      DO 20 I = 2, NI
         BFJ(1) = BFI(I)
         DO 10 J = 2, NJ
            BFJ(J) = XX*BFJ(J)+YY*BFJ(J-1)
            CDFX = CDFX+POI(I)*POJ(J)*BFJ(J)
   10    CONTINUE
   20 CONTINUE
      RETURN
      END
```