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

Charles P. Reeve

Let $X_1$ and $X_2$ be independent noncentral chi-squared random variables with degrees of freedom $\nu_1$ and $\nu_2$ (both $>0$) and noncentrality parameters $\lambda_1$ and $\lambda_2$ (both $>0$) respectively. Then the random variable

$$Y = \left( \frac{X_1}{\nu_1} \right) / \left( \frac{X_2}{\nu_2} \right)$$

(1)

is said to have the doubly noncentral F distribution, indicated by $Y \sim F'(\nu_1, \nu_2; \lambda_1, \lambda_2)$. This distribution has been used in the evaluation of the power function of analysis of variance tests in which interaction or bias effects occur, as in Scheffe [11]. Numerical examples of this usage are given in Bulgren [2] and Tiku [13]. It has also been used in engineering problems in the context of information theory as discussed in Price [8].


The author has been unable to find any published algorithms for computing exact values of the $F'$ c.d.f. although computer programs have obviously been used in generating published tables in [2,5,12,14]. The purpose of this note is to present an efficient algorithm for computing the $F'$ c.d.f. to a specified accuracy using exact formulas.

The algorithm uses the series representation in eq. (2.2) of [2] which, unfortunately, contains typographical errors. With these corrected the c.d.f. of $Y$ can be re-written

$$F_Y(x) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} A_i B_j I(u, \nu_1/2+i, \nu_2/2+j)$$

(2)

where $A_i = (\lambda_1/2)^{i/2} e^{-\lambda_1/2} / \Gamma(i+1)$, $B_j = (\lambda_2/2)^{j/2} e^{-\lambda_2/2} / \Gamma(j+1)$, $u = \nu_1 x / (\nu_1 x + \nu_2)$,

and $x>0$. The $A_i$ and $B_j$ 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
\( B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1} \, dt \). For computational purposes the two infinite series must be truncated, thus (2) is re-expressed as

\[
F_Y(x) = \sum_{i=I'}^{I''} \sum_{j=J'}^{J''} A_i B_j I(u, \nu_1/2+i, \nu_2/2+j) + R
\]

where \( I', I'', J', \) 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 \( I', I'', J', \) and \( J'' \) such that \( \sum_{i=I'}^{I''} A_i > 1-\varepsilon/2 \) and \( \sum_{j=J'}^{J''} B_j > 1-\varepsilon/2 \) yields \( R < \varepsilon \) provided \( \varepsilon > 0 \). Therefore, \( \varepsilon \) 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 \( i \) and \( j \) over the largest of the Poisson probabilities \( A_i \) and \( B_j \) respectively. The final task is to compute the \( (I''-I'+1)(J''-J'+1) \) beta c.d.f.'s and the double summation. An efficient procedure for doing this is to compute only \( I(u, \nu_1/2+I', \nu_2/2+J') \) and \( I(u, \nu_1/2+I', \nu_2/2+J') \) directly, indicated by the symbols "x" and "y" in figure 1. The remaining beta c.d.f.'s are computed using the recurrence relations

\[
I(x,a,b) = I(x,a,b+1) - x^a(1-x)^b / [bB(a,b)], \quad (4a)
\]

\[
I(x,a,b) = I(x,a+1,b) + x^a(1-x)^b / [aB(a,b)], \quad (4b)
\]

\[
I(x,a,b) = xI(x,a-1,b) + (1-x)I(x,a,b-1) \]

as found in Abramowitz and Stegun [1]. Subject to the restrictions \( J' < J' \leq J'' \)

![Figure 1](image_url)

Computation of the \( I(u, \nu_1/2+i, \nu_2/2+j) \)
and $I' \leq I \leq I''$, $J'$ and $I'$ are chosen to maximize the magnitudes of the rightmost terms in (4a) and (4b) 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 $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$. 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 figure 1 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.

The above algorithm has been incorporated into the FORTRAN subroutine CDFDNF. If $x < 0$ then 0 is returned. 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 CDFDNF these routines are the subroutine CDFBET and the function GAMLOG as described in Reeve [9,10]. Other routines which can be substituted for CDFBET and GAMLOG are those in [3,6,7]. For all practical purposes the absolute error criterion $\varepsilon$ will be met if the beta c.d.f. routine is accurate to two or three digits beyond $\varepsilon$.

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 $\lambda_1$ and $\lambda_2$ become large. This is illustrated in table 1 where, for selected parameter values, the CPU time for this method is compared with that of the "brute force" approach in which each beta c.d.f. is directly evaluated by a subroutine call. The computing time and storage also increase as $\varepsilon$ becomes small, but are unaffected by $\nu_1$ and $\nu_2$. The computations in table 1 were done on the CDC Cyber 180/855 computer at NBS.

The current dimension limits in CDFDNF allow values of both $\lambda_1$ and $\lambda_2$ up to about 10,000 with $\varepsilon$ as small as 10^{-8}, but these limits could easily be increased by the user. The limiting factor in using CDFDNF 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 figure 1 are actually stored. The Poisson probabilities $A_i$, ..., $A_{11}$ and $B_j$, ..., $B_j'$ are also stored.

If $\lambda_2 = 0$ then the doubly noncentral $F$ reduces to the (singly) noncentral $F$, and if $\lambda_1 = \lambda_2 = 0$ it reduces to the central $F$. In either case, CDFDNF will run almost as efficiently as routines designed for those specific cases.

Portions of tables in [2,5,14] were reproduced by CDFDNF and agreed to within roundoff error in each case.

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

```
CALL CDFDNF(X,DF1,DF2,ALAMB1,ALAMB2,EPS,IFLAG,DFX)
```
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 CDFDNF 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 \( F^{\prime}(v_1, v_2, \lambda_1, \lambda_2) \) using both CDFDNF and the "brute force" method (described in the text), for selected parameter values.

<table>
<thead>
<tr>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( \lambda_1 )</th>
<th>( \lambda_2 )</th>
<th>( x )</th>
<th>( P(F^{\prime} \leq x) )</th>
<th>CDFDNF</th>
<th>&quot;Brute Force&quot;</th>
<th>CPU seconds</th>
<th>No. beta c.d.f. values</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>2.0</td>
<td>0.757918</td>
<td>&lt;0.01</td>
<td>0.13</td>
<td>196</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>25</td>
<td>2.0</td>
<td>0.997561</td>
<td>&lt;0.01</td>
<td>0.30</td>
<td>476</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>25</td>
<td>5</td>
<td>2.0</td>
<td>0.190910</td>
<td>&lt;0.01</td>
<td>0.31</td>
<td>476</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>25</td>
<td>25</td>
<td>2.0</td>
<td>0.897835</td>
<td>&lt;0.01</td>
<td>0.69</td>
<td>1,156</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>2.0</td>
<td>0.593795</td>
<td>&lt;0.01</td>
<td>0.14</td>
<td>196</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>25</td>
<td>2.0</td>
<td>0.943093</td>
<td>&lt;0.01</td>
<td>0.30</td>
<td>476</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>25</td>
<td>5</td>
<td>2.0</td>
<td>0.026209</td>
<td>&lt;0.01</td>
<td>0.28</td>
<td>476</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>25</td>
<td>25</td>
<td>2.0</td>
<td>0.289601</td>
<td>&lt;0.01</td>
<td>0.73</td>
<td>1,156</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>2.0</td>
<td>0.898330</td>
<td>&lt;0.01</td>
<td>0.12</td>
<td>196</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>5</td>
<td>25</td>
<td>2.0</td>
<td>0.999879</td>
<td>&lt;0.01</td>
<td>0.26</td>
<td>476</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>25</td>
<td>5</td>
<td>2.0</td>
<td>0.657879</td>
<td>&lt;0.01</td>
<td>0.29</td>
<td>476</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>25</td>
<td>25</td>
<td>2.0</td>
<td>0.997703</td>
<td>&lt;0.01</td>
<td>0.66</td>
<td>1,156</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>2.0</td>
<td>0.868071</td>
<td>&lt;0.01</td>
<td>0.14</td>
<td>196</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>5</td>
<td>25</td>
<td>2.0</td>
<td>0.998234</td>
<td>&lt;0.01</td>
<td>0.30</td>
<td>476</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>25</td>
<td>5</td>
<td>2.0</td>
<td>0.367101</td>
<td>&lt;0.01</td>
<td>0.32</td>
<td>476</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>25</td>
<td>25</td>
<td>2.0</td>
<td>0.934321</td>
<td>&lt;0.01</td>
<td>0.71</td>
<td>1,156</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>80</td>
<td>80</td>
<td>1.1</td>
<td>0.552328</td>
<td>0.03</td>
<td>2.55</td>
<td>3,969</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>400</td>
<td>400</td>
<td>1.1</td>
<td>0.582507</td>
<td>0.12</td>
<td>16.28</td>
<td>20,164</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>2000</td>
<td>2000</td>
<td>1.1</td>
<td>0.664981</td>
<td>0.59</td>
<td>108.10</td>
<td>101,124</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>10000</td>
<td>10000</td>
<td>1.1</td>
<td>0.825080</td>
<td>2.68</td>
<td>800</td>
<td>505,521</td>
<td></td>
</tr>
<tr>
<td>*14</td>
<td>15</td>
<td>50000</td>
<td>50000</td>
<td>1.1</td>
<td>0.981351</td>
<td>13.76</td>
<td>-</td>
<td>2,528,100</td>
<td></td>
</tr>
</tbody>
</table>

* required a doubling of current dimension limits
References


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


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


SUBROUTINE CDFDNF (X,DF1,DF2,ALAMB1,ALAMB2,_EPS,IFLAG,CDFX)

FOR: COMPUTING THE CUMULATIVE DISTRIBUTION FUNCTION OF THE DOUBLY NONCENTRAL F DISTRIBUTION TO A SPECIFIED ACCURACY (TRUNCATION ERROR IN THE INFINITE SERIES REPRESENTATION GIVEN BY EQUATION 2.2 IN REFERENCE 1 BELOW). THE BETA C.D.F. ROUTINE IS CALLED AT MOST TWO TIMES. FURTHER VALUES OF THE BETA C.D.F. ARE OBTAINED FROM RECURRENCE RELATIONS GIVEN IN REFERENCE 2. REFERENCE 3 GIVES A DETAILED DESCRIPTION OF THE ALGORITHM HEREIN.

THIS PROGRAM MAY ALSO BE EFFICIENTLY USED TO COMPUTE THE CUMULATIVE DISTRIBUTION FUNCTIONS OF THE SINGLY NONCENTRAL AND CENTRAL F DISTRIBUTIONS BY SETTING THE APPROPRIATE NONCENTRALITY PARAMETERS EQUAL TO ZERO.

CHECKS ARE MADE TO ASSURE THAT ALL PASSED PARAMETERS ARE WITHIN VALID RANGES AS GIVEN BELOW. NO UPPER LIMIT IS SET FOR THE NONCENTRALITY PARAMETERS, BUT VALUES UP TO ABOUT 10,000 CAN BE HANDLED WITH THE CURRENT DIMENSION LIMITS. THE COMPUTED VALUE CDFX IS VALID ONLY IF IFLAG=0 ON RETURN.

NOTE: IN EQUATION 2.2 OF REFERENCE 1 THE AUTHOR HAS MISTAKENLY REVERSED THE ARGUMENTS OF THE INCOMPLETE BETA FUNCTION. THEY SHOULD READ \((M/2+R)-(N/2+S)\) WHERE M AND N ARE THE DEGREES OF FREEDOM ASSOCIATED WITH THE NUMERATOR AND DENOMINATOR RESPECTIVELY OF THE F STATISTIC. TO FURTHER CONFUSE THE ISSUE, THE AUTHOR HAS REVERSED THE USAGE OF M AND N IN SECTION 1 OF THE PAPER.

NOTE: IN SUBROUTINE EDGEF THE DOUBLE PRECISION CONSTANT DEUFLO IS THE EXPONENTIAL UNDERFLOW LIMIT WHOSE CURRENT VALUE IS SET AT \(-6908\). ON A COMPUTER WHERE \(10^{-13}\) CAUSES UNDERFLOW THIS LIMIT SHOULD BE CHANGED.

SUBPROGRAMS CALLED: CDFBET (BETA C.D.F.)
GAMLLOG (DOUBLE PRECISION LOG OF GAMMA FUNCTION)
POISSF, EDGEF (ATTACHED)

CURRENT VERSION COMPLETED SEPTEMBER 29, 1988

REFERENCES:

DEFINITION OF PASSED PARAMETERS:
• X = VALUE \(\geq 0\) AT WHICH THE C.D.F. IS TO BE COMPUTED (REAL)
• DF1 = DEGREES OF FREEDOM \(>0\) IN THE NUMERATOR (REAL)
• DF2 = DEGREES OF FREEDOM \(>0\) IN THE DENOMINATOR (REAL)
• ALAMB1 = THE NONCENTRALITY PARAMETER \(\geq 0\) FOR THE NUMERATOR (REAL) [EQUAL TO ZERO FOR THE CENTRAL F DISTRIBUTION]
• ALAMB2 = THE NONCENTRALITY PARAMETER \(\geq 0\) FOR THE DENOMINATOR (REAL) [EQUAL TO ZERO FOR THE SINGLY NONCENTRAL F AND CENTRAL F DISTRIBUTIONS]
• EPS = THE DESIRED ABSOLUTE ACCURACY OF THE C.D.F. (REAL) \([10^{-10} = EPS \geq 10^{+(-10)}]\)
• IFLAG = ERROR INDICATOR ON OUTPUT (INTEGER) INTERPRETATION:
0 -> NO ERRORS DETECTED
1, 2 -> ERROR FLAGS FROM SUBROUTINE CDFBET
3 -> EITHER ALAMB1 OR ALAMB2 IS < 0
4 -> EITHER DF1 OR DF2 IS <= 0
5 -> EPS IS OUTSIDE THE RANGE [10**(-10), 1]
6 -> VECTOR DIMENSIONS ARE TOO SMALL - INCREASE NX

CDFX = THE DOUBLY NONCENTRAL F C.D.F. EVALUATED AT X (REAL)

* INDICATES PARAMETERS REQUIRING INPUT VALUES

PARAMETER (NX=1000)
DIMENSION BFI(NX),BFJ(NX),POI(NX),POJ(NX)
CDFX = 0.0

C CHECK VALIDITY OF ARGUMENTS
IF (ALAMB1.LE.0.0.OR.ALAB2.LE.0.0) THEN
  IFLAG = 3
  RETURN
ENDIF
IF (DF1.LE.0.0.OR.DF2.LE.0.0) THEN
  IFLAG = 4
  RETURN
ENDIF
IF (EPS.GT.1.0.OR.EPS.LT.1.0E-10) THEN
  IFLAG = 5
  RETURN
ENDIF
ENDIF
IFLAG = 0

C SET ERROR CRITERION FOR THE BETA C.D.F. (PECULIAR TO CDFBET)
EPS3 = 0.001*EPS

FA = 0.5*ALAMB1
GA = 0.5*ALAMB2
FB = 0.5*DF1
GB = 0.5*DF2
YY = DF2/(DF2+DF1*X)
IF (YY.GE.1.0) RETURN
XX = 1.0-YY
IF (XX.GE.1.0) THEN
  CDFX = 1.0
  RETURN
ENDIF

C COMPUTE POISSON PROBABILITIES IN VECTORS POI AND POJ
CALL POISSF (FA,EPS,IMIN,NI,POI,NX,IFLAG)
IF (IFLAG.NE.0) RETURN
FD = FB*REAL(IMIN)
CALL POISSF (GA,EPS,MIN,NJ,POJ,NX,IFLAG)
IF (IFLAG.NE.0) RETURN
GC = GB*REAL(JMIN)

C COMPUTE BETA C.D.F. BY RECURRENCE WHEN I=IMIN AND J=JMIN TO JMAX
CALL EDFGEF (NJ,GC,FC,YY,XX,BFJ,CDFX,POJ,POI,EPS3,IFLAG,1)
IF (NI.LE.1.OR.IFLAG.NE.0) RETURN

C COMPUTE BETA C.D.F. BY RECURRENCE WHEN J=JMIN AND I=IMIN TO IMAX
BFI(1) = BFJ(1)
CALL EDFGEF (NI,FC,GC,XX,YY,BFI,CDFX,POJ,POI,EPS3,IFLAG,2)
IF (NJ.LE.1.OR.IFLAG.NE.0) RETURN

C COMPUTE BETA C.D.F. BY RECURRENCE WHEN I>IMIN AND J>JMIN
DO 20 I = 2, NI
  BFJ(I) = 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
SUBROUTINE POISSF (ALAMB, EPS, L, NSPAN, V, NV, IFLAG)

C COMPUTE THE POISSON(ALAMB) PROBABILITIES OVER THE RANGE [L,K]
C WHERE THE TOTAL TAIL PROBABILITY IS LESS THAN EPS/2. 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 = 1.800-0.500*DBLE(EPS)
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.DLIM) GOTO 20
  PK = ALAMB*PK/REAL(K+1)
ELSE
  NL = NL-1
  V(NL) = PL
  DSUM = DSUM+DBLE(PL)
  L = L-1
  IF (DSUM.GE.DLIM) GOTO 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

SUBROUTINE EDGEF (NK, FC, GC, XX, YY, BFK, CDFX, POI, POJ, EPS3, IFLAG, L)

C COMPUTE THE BETA C.D.F.'S BY A RECURRENCE RELATION ALONG THE EDGES
C I = JMIN AND J = JMIN OF A GRID. THE CORRESPONDING COMPONENTS OF
C THE F" 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 / -69.000 /
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

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(I)*BFK(I)
50 CONTINUE
RETURN
END