Algorithms

J. G. HERRIOT, Editor

ALGORITHM 352 CHARACTERISTIC VALUES AND ASSOCIATED SOLUTIONS OF MATHIEU'S DIFFERENTIAL EQUATION [S22]

Donald S. Clemm (Recd. 2 June 1967, 18 Apr. 1968, 6 Jan. 1969 and 10 Mar. 1969)

Aerospace Research Laboratories Wright-Patterson Air Force Base OH 45433

KEY WORDS AND PHRASES: Mathieu's differential equation, Mathieu function, characteristic value, periodic solution, radial solution

CR CATEGORIES: 5.12

Comments Algorithm 352 is a package of double-precision Fortran routines which consists of the following primary routines:

MFCVAL—referred to as Algorithm 352 (Part A)

MATH—referred to as Algorithm 352 (Part B)

BESSEL—referred to as Algorithm 352 (Part C)

MFCVAL computes characteristic values of Mathieu's differential equation. MATH computes the associated solutions of this equation, using BESSEL as an auxiliary routine to evaluate Bessel functions. This latter routine may be used independently.

There are other, secondary routines included in the package, and the numbering system (e.g. Algorithm 352 (Part A.1)) indicates somewhat the mutual relation between them, as well as their relation to the primary routines. The functioning of the routines and the linkages between them are explained in the comments prefacing each one. All literature citations refer to the following list.

References:

1. Abramowitz, M., and Stegun, I. A. (Eds.). *Handbook of Mathematical Functions.* NBS Appl. Math. Ser. 55, US Govt. Print. Off., Washington, D.C., 1964.
2. Blanch, G. Numerical evaluation of continued fractions. *SIAM Rev. 6*, 4 (1964), 383–421.
3. Blanch, G. Numerical aspects of Mathieu eigenvalues. *Rend. Circ. Mat. Palermo* (2) *15* (1966), 51–97.
4. Blanch, G., and Clemm, D. S. *Tables Relating to the Radial Mathieu Functions, Vol. 1, Functions of the First Kind.* US Govt. Print. Off., Washington, D.C., 1962.
5. Blanch, G., and Clemm, D. S. *Tables Relating to the Radial Mathieu Functions, Vol. 2, Functions of the Second Kind.* US Govt. Print. Off., Washington, D.C., 1965.
6. Ince, E. L. Tables of the elliptic cylinder functions. *Proc. Roy. Soc. Edinburgh 52* (1932), 355–423; also Zeros and turning points. *Proc. Roy. Soc. Edinburgh 52* (1932), 424–433.
7. National Bureau of Standards. *Tables Relating to Mathieu Functions.* Appl. Math. Ser. 59, US Govt. Print. Off., Washington, D.C., 1967. (second ed.)
8. Stratton, J. A., Morse, P. M., Chu, L. J., and Hutner, R. A. *Elliptic Cylinder and Spheroidal Wave Functions.* Wiley, New York, 1941.

Algorithm 352 (Part A) MFCVAL (Characteristic Values)

Comments The subrountine MFCVAL computes the first N characteristic values, a, together with upper and lower bounds, of Mathieu's differential equation for nonnegative values of the real parameter, q. The equation can be written in the form

$$y'' + (a - 2q \cos 2x)y = 0, \qquad (1)$$

where $a = a_r$ ($a = b_r$) indicates a characteristic value associated with the even (odd) periodic solutions.

The method consists of three steps: (1) calculate a rough approximation based on coefficients obtained from curve-fitting of available tabulations, (2) determine crude upper and lower bounds, and (3) iterate, using a variation of Newton's method. For a justification of this method, see [3].

Explanation of the arguments:

- N — the given number of characteristic values desired
- R — given as N−1 or N according as the characteristic values are to be associated with the even or odd solutions, respectively
- QQ — the given nonnegative parameter q
- CV — the computed 6 by N array of characteristic values and bounds
- J — the number of characteristic values successfully computed. J $\neq$ N indicates that J values were computed with an error occurring on the J + 1 value. A printed message will accompany such an error condition.

The output array, CV, must be appropriately dimensioned in the calling program and upon return will contain the following data:

For the Kth characteristic value, K = 1, 2, $\cdots$, J,

- CV (1, K) — the characteristic value a
- CV (2, K) — the function $D(a) = -T_m(a)/T_m'(a)$
- CV (3, K) — a_L , a lower bound of a
- CV (4, K) — the function $D(a_L)$
- CV (5, K) — a_U , an upper bound of a
- CV (6, K) — the function $D(a_U)$.

Reference is again made to [3], where the function $T_m(a)$ is defined and it is proved that $T_m(a) = 0$ if and only if a is a characteristic value. From this, it can be said that the function D is an indication of the accuracy of its argument, since $a + D(a)$ would be the value of the next iteration.

The first executable statement in MFCVAL sets a tolerance of 10^{-13}. This may be changed by the user, but the following comments should be heeded if it is attempted.

If it is desired to reduce the tolerance in order to achieve the greatest possible accuracy, care should be taken that the tolerance is not less than $10^{-(n-2)}$ when executing the routines on a machine which uses n-digit arithmetic. In other words, if the user's computer employs 24-digit arithmetic, this tolerance should be no less than 10^{-22}. A too small tolerance will impose an unattainable accuracy requirement and overflow may occur.[1]

On the other hand, some time-saving may be achieved, at the expense of accuracy, by making the tolerance less stringent. A tolerance of 10^{-d} will produce results good to at least d digits. This is a conservative estimate, since one additional iteration is performed after the tolerance is met and, normally, the convergence of successive iterations is quadratic.

Perhaps it should be noted again that the accuracy of any characteristic value, a, can be determined from the size of it relative to the function $D(a)$. See the description of the

[1] The constant in statement numbers 425 and 445 is introduced to avoid the possibility of a zero tolerance. This should not be altered unless the routines are being run on a machine which uses arithmetic of more than 16 digits, and then it must not be less than $10^{-(n-2)}$, with n defined as above.

contents of the output array CV. MFCVAL calls on the subroutines:

BOUNDS—referred to as Algorithm 352 (Part A.1)
MFITR8—referred to as Algorithm 352 (Part A.2)
TMOFA—referred to as Algorithm 352 (Part A.3)

```
      SUBROUTINE MFCVAL (N,R,QQ,CV,J)
C     ****************

      INTEGER
     *       J,K,KK,L,M,N,R,TYPE

      DOUBLE PRECISION
     *       A,CV,DL,DR,DTM,Q,QQ,
     *       T,TM,TOL,TOLA

      DIMENSION
     *       CV(6,N)

      EQUIVALENCE
     *       (DL,DR,T)

      COMMON /MF1/
     *       Q,TOL,TYPE,DUMMY(4)

      TOL = 1.D-13

      IF (N-R) 10,10,20

   10 L    = 1
                                      GO TO 30
   20 L    = 2
   30 Q    = QQ
      DO 500 K = 1,N
         J     = K

         IF (Q) 960,490,40

   40    KK    = MIN0(K,4)
         TYPE  = 2*MOD(L,2)+MOD(K-L+1,2)
C     FIRST APPROXIMATION
         GO TO (100,200,300,400), KK

  100    IF (Q-1.D0) 110,140,140

  110    GO TO (120,130), L
  120    A     = 1.D0-Q-.125D0*Q*Q
                                      GO TO 420
  130    A     = Q*Q
         A     = A*(-.5D0+.0546875D0*A)
                                      GO TO 420

  140    IF (Q-2.D0) 150,180,180

  150    GO TO (160,170), L
  160    A     = 1.033D0-1.0746D0*Q-
     *           .0688D0*Q*Q
                                      GO TO 420
  170    A     = .23D0-.495D0*Q-
     *           .191D0*Q*Q
                                      GO TO 420
  180    A     = -.25D0-2.D0*Q+
     *           2.D0*DSQRT(Q)
                                      GO TO 420

  200    DL    = L

         IF (Q*DL-6.D0) 210,350,350

  210    GO TO (220,230), L
  220    A     = 4.01521D0-Q*
     *           (.046D0+.0667857D0*Q)
                                      GO TO 420
  230    A     = 1.D0+1.05007D0*Q-
     *           .180143D0*Q*Q
                                      GO TO 420

  300    IF (Q-8.D0) 310,350,350

  310    GO TO (320,330), L
  320    A     = 8.93867D0+.178156D0*Q-
     *           .0252132D0*Q*Q
                                      GO TO 420
  330    A     = 3.70017D0+.953485D0*Q-
     *           .0475065D0*Q*Q
                                      GO TO 420

  350    DR    = K-1
         A     = CV(1,K-1)-DR+
     *           4.D0*DSQRT(Q)
                                      GO TO 420

  400    A     = CV(1,K-1)-CV(1,K-2)
         A     = 3.D0*A+CV(1,K-3)

  420    IF (Q.GE.1.D0) GO TO 440

         IF (K.NE.1) GO TO 430

  425    TOLA  = DMAX1(DMIN1(TOL,DABS(A))
     *           ,1.D-14)
                                      GO TO 450
  430    TOLA  = TOL*DABS(A)
                                      GO TO 450
  440    TOLA  = TOL*DMAX1(Q,DABS(A))
  445    TOLA  = DMAX1(DMIN1(TOLA,DABS(A)
     *           ,.4D0*DSQRT(Q))
     *           ,1.D-14)
C     CRUDE UPPER AND LOWER BOUNDS
  450    CALL BOUNDS (K,A,TOLA,CV,N,M)

         IF (M.NE.0)
     *      IF (M-1) 470,910,900

C     ITERATE
         CALL MFITR8 (TOLA,CV(1,K),
     *                CV(2,K),M)

         IF (M.GT.0) GO TO 920

C     FINAL BOUNDS AND FUNCTIONS, D
  470    T        = CV(1,K)-TOLA
         CALL TMOFA (T,TM,DTM,M)

         IF (M.GT.0) GO TO 940

         CV(3,K) = T
         CV(4,K) = -TM/DTM
  480    T        = CV(1,K)+TOLA
         CALL TMOFA (T,TM,DTM,M)

         IF (M.GT.0) GO TO 950

         CV(5,K) = T
         CV(6,K) = -TM/DTM
                                      GO TO 500
C     Q EQUALS ZERO
  490    CV(1,K) = (K-L+1)**2
         CV(2,K) = 0.D0
         CV(3,K) = CV(1,K)
         CV(4,K) = 0.D0
         CV(5,K) = CV(1,K)
         CV(6,K) = 0.D0
  500 CONTINUE
  550                                 RETURN
C     PRINT ERROR MESSAGES
  900 WRITE (6,901) K
  901 FORMAT(20H0CRUDE BOUNDS CANNOT,
     *       22H BE LOCATED, NO OUTPUT,
     *       7H FOR K=I2)
                                      GO TO 930
  910 WRITE (6,911) K
  911 FORMAT(20H0ERROR IN SUBPROGRAM,
     *       22H TMOFA, VIA SUBPROGRAM,
     *       18H BOUNDS, NO OUTPUT,
     *       7H FOR K=I2)
                                      GO TO 930
  920 WRITE (6,921) K
  921 FORMAT(20H0ERROR IN SUBPROGRAM,
     *       22H TMOFA, VIA SUBPROGRAM,
     *       18H MFITR8, NO OUTPUT,
     *       7H FOR K=I2)
  930 J      = J-1
                                      GO TO 550
  940 WRITE (6,941) K
  941 FORMAT(20H0ERROR IN SUBPROGRAM,
     *       22H TMOFA, NO LOWER BOUND,
     *       7H FOR K=I2)
      CV(3,K) = 0.D0
      CV(4,K) = 0.D0
                                      GO TO 480
  950 WRITE (6,951) K
  951 FORMAT(20H0ERROR IN SUBPROGRAM,
     *       22H TMOFA, NO UPPER BOUND,
     *       7H FOR K=I2)
      CV(5,K) = 0.D0
      CV(6,K) = 0.D0
                                      GO TO 500
  960 WRITE (6,961)
  961 FORMAT(20H0Q GIVEN NEGATIVELY,,
     *       20H USED ABSOLUTE VALUE)
      Q      = -Q
                                      GO TO 40
      END
```

Algorithm 352 (Part B)
MATH (Mathieu Functions)

Comments The subroutine MATH computes various solutions (and their derivatives), of either Mathieu's differential equation or Mathieu's modified equation, which are associated with the characteristic values.

The *even periodic* solution of equation (1) is

$$ce_r(x,q) = \sum_{k=0}^{\infty} A_{2k+p} \cos (2k+p)x, \quad (2)$$

associated with $a_r(q)$, and the *odd periodic* solution is

$$se_r(x,q) = \sum_{k=0}^{\infty} B_{2k+p} \sin (2k+p)x, \quad (3)$$

associated with $b_r(q)$. The order, r, is of the form $2n + p$. The n is a nonnegative integer while $p = 0$ or 1 indicates the solution is of period π or 2π. Calculation of the periodic solutions allows the following three options of normalization:

(a) Neutral. We define *neutral* coefficients such that $\bar{A}_{2k+p} = A_{2k+p}/A_{2s+p}$, where s is chosen so that A_{2s+p} is the numerically largest one of the set. The $\bar{B}_{2k+p}$ are similarly defined. This has the computationally convenient effect of making the largest coefficient equal to unity, hence all calculations are carried out with them. If a normalization other than *neutral* is selected, it is effected on the output array F only, the coefficients themselves remaining unchanged.

(b) Ince. The normalization adopted in [6] is defined so that if $y(x,q)$ represents either function (2) or (3) then

$$\int_0^{2\pi} y^2(x, q)dx = \pi.$$

(c) Stratton. As defined in [8], and in the notation of [7], this normalization is effected so that

$$Se_r(q, 0) = \left[\frac{d}{dx} So_r(q, x)\right]_{x=0} = 1,$$

where Se is the even solution and So the odd.

If we replace x by ix in (1), we get

$$y'' - (a-2q \cosh 2x)\, y = 0, \quad (4)$$

known as Mathieu's modified equation. The solutions of (4) have been termed *radial* in [8] and, for characteristic values, can be put in the following form, using the notation of [4] and [5]:

$$Mc_r^{(i)}(x,q) = \sum_{k=0}^{\infty} (-1)^{n+k} A_{2k+p} [F_k + G_k]/A_{2s+p}\epsilon_{2s+p}, \quad (5)$$

associated with $a_r(q)$, and

$$Ms_r^{(i)}(x,q) = \sum_{k=0}^{\infty} (-1)^{n+k} B_{2k+p} [F_k - G_k]/B_{2s+p}, \quad (6)$$

associated with $b_r(q)$. The order r equals $2n + p$, as in (2) and (3), and $\epsilon_m = 1$ if $m \neq 0$, but $\epsilon_0 = 2$. The choice of s is arbitrary here, but for numerical purposes we choose it in the manner described previously for *neutral* normalization. The coefficients are the same as defined in (2) and (3), while F_k and G_k involve the Bessel functions as follows:

$$F_k = J_{k-s}(u_1)\, Z^{(j)}_{k+p+s}(u_2), \tag{7}$$

$$G_k = J_{k+p+s}(u_1)\, Z^{(j)}_{k-s}(u_2), \tag{8}$$

$$u_1 = q^{\frac{1}{2}}e^{-x}, \qquad u_2 = q^{\frac{1}{2}}e^{x},$$

$$Z^{(1)}_m(u) = J_m(u),\ Z^{(2)}_m(u) = Y_m(u).$$

The solutions (5)–(6) are said to be of the first or second kind depending on whether $j = 1$ or 2 in (5)–(8).

Explanation of the arguments:

- XX — the given independent variable x
- QQ — the given positive parameter q
- R — the given order r
- CV — the given characteristic value, $a_r(q)$ or $b_r(q)$
- SOL — given as 1, 2, or 3 according as the desired solution is (1) radial of the first kind, (2) radial of the second kind, or (3) periodic
- FNC — given as 1, 2, 3, or 4 according as the desired solution is (1) associated with b_r, (2) associated with a_r, (3) the derivative of solution (1), or (4) the derivative of solution (2)
- NORM — given as 1, 2, or 3 according as the desired normalization is (1) defined as *neutral*, (2) defined by Ince, or (3) defined by Stratton. (This argument is decoded only if SOL = 3.)
- F — the computed three-element array, containing: (1) the solution value, (2) the series term of largest magnitude, and (3) the last term included in the summation
- K — the computed two-element array, containing: (1) the index, k, of the term in F(2), and (2) the index of the term in F(3)
- M — the error indicator cell: M = 0 indicates successful execution of subprogram, M = 1 signifies an error condition explained by an accompanying printed message.

The accuracy of results (within limits) and the speed of convergence may be altered by the user. See SUM (Algorithm 352 (Part B.2)) for details.

MATH calls on the subroutines:

COEF—referred to as Algorithm 352 (Part B.1)

SUM—referred to as Algorithm 352 (Part B.2)

BESSEL—referred to as Algorithm 352 (Part C)

```
      CUBROUTINE MATH (XX,QQ,R,CV,SOL,
     *                 FNC,NORM,F,K,M)
C     ***************
      INTEGER
     *        FNC,I,K(2),KLAST,KMAX,L,
     *        LL,M,MF,ML,MM,MO,M1,M2S,
     *        N,NORM,P,R,S,SOL,TYPE
      DOUBLE PRECISION
     *        A,AB,CV,DLAST,DMAX,F(3),G,
     *        J,Q,QQ,T,TOL,U1,U2,X,XX,Y
      EXTERNAL
     *        DC,DDC,DDS,DS,DPC,DPS,
     *        PC,PS
      COMMON
     *        J(250),Y(250),U1,U2,N,P,S,
     *        L,X,T,I,LL,G,DMAX,DLAST,
     *        KMAX,KLAST,DUM1(578),A,
     *        DUM2(6),MM,ML,AB(200)
      COMMON /MF1/
     *        Q,TOL,TYPE,M1,MO,M2S,MF
      M     = 0
      IF (SOL.LT.1 .OR.
     *    SOL.GT.3 .OR.
     *    FNC.LT.1 .OR.
     *    FNC.GT.4) GO TO 400
      A     = CV
      Q     = QQ
      TOL   = 1.D-13
      TYPE  = 2*MOD(FNC,2)+MOD(R,2)
      CALL COEF (M)
      IF (M) 410,10,420
   10 N     = R/2
      P     = MOD(R,2)
      S     = MM/2
      L     = ML/2
      X     = XX
      T     = 1.D0
      IF (SOL.EQ.3)
     *   GO TO (150,160,170,180), FNC
      U1    = DSQRT(Q)*DEXP(-X)
      U2    = Q/U1
      LL    = L+S+P
C     COMPUTE BESSEL FUNCTIONS
      CALL BESSEL (1,U1,J,LL)
      CALL BESSEL (SOL,U2,Y,LL)
C     EVALUATE SELECTED FUNCTION
      GO TO (50,60,70,80), FNC
   50 CALL SUM (DS)
                                GO TO 300
   60 CALL SUM (DC)
                                GO TO 300
   70 CALL SUM (DDS)
                                GO TO 300
   80 CALL SUM (DDC)
                                GO TO 300
  150 CALL SUM (PS)
                                GO TO 200
  160 CALL SUM (PC)
                                GO TO 200
  170 CALL SUM (DPS)
                                GO TO 200
  180 CALL SUM (DPC)
  200 IF (NORM-2) 300,210,250
C     INCE NORMALIZATION
  210 T     = AB(1)**2
      IF (TYPE.EQ.0) T = T+T
      DO 220 I = 1,L
         T     = T+AB(I+1)**2
  220 CONTINUE
      T     = DSQRT(T)
      I     = MO/2
      IF (AB(I).LT.0.D0) T = -T
                                GO TO 300
C     STRATTON NORMALIZATION
  250 IF (TYPE.GT.1) GO TO 270
      T     = AB(1)
      DO 260 I = 1,L
         T     = T+AB(I+1)
  260 CONTINUE
                                GO TO 300
  270 T     = DBLE(FLOAT(P))*AB(1)
      DO 280 I = 1,L
         T     = T+AB(I+1)*
     *           DBLE(FLOAT(2*I+P))
  280 CONTINUE
  300 F(1)  = G/T
      F(2)  = DMAX/T
      F(3)  = DLAST/T
      K(1)  = KMAX
      K(2)  = KLAST
  350                           RETURN
C     PRINT ERROR MESSAGES
  400 WRITE (6,401)
  401 FORMAT(18H0SOL OR FNC OUT OF,
     *       17H RANGE, NO OUTPUT)
                                GO TO 450
  410 WRITE (6,411)
  411 FORMAT(15H0MORE THAN 200 ,
     *       22HCOEFFICIENTS REQUIRED,,
     *       20H QQ AND R TOO LARGE,,
     *       10H NO OUTPUT)
                                GO TO 450
  420 WRITE (6,421)
  421 FORMAT(20H0ERROR IN SUBPROGRAM,
     *       22H TMOFA, VIA SUBPROGRAM,
     *       13H COEF, VERIFY,
     *       21H ARGUMENTS, NO OUTPUT)
  450 M     = 1
      F(1)  = 0.D0
      F(2)  = 0.D0
      F(3)  = 0.D0
      K(1)  = 0
      K(2)  = 0
                                GO TO 350
      END
```

Algorithm 352 (Part A.1)
BOUNDS (Crude Bounds)
(Called by MFCVAL)

Comments The subroutine BOUNDS determines crude upper and lower bounds for the Kth characteristic value, K ≦ N.

Explanation of the other arguments:

- APPROX — the first approximation
- TOLA — the tolerance determined by subroutine MFCVAL
- CV — the 6 by N array described in subroutine MFCVAL
- N — variable dimension of the CV array
- MM — an indicator cell used to communicate unusual and error conditions to subroutine MFCVAL

The output, $a_0 < a < a_1$, is put into the common block labeled MF2.

BOUNDS calls on the subroutine:

TMOFA—referred to as Algorithm 352 (Part A.3)

```
      SUBROUTINE BOUNDS (K,APPROX,
     *                   TOLA,CV,N,MM)
C     ****************
      INTEGER
     *        K,KA,M,MM,N
      DOUBLE PRECISION
     *        A,APPROX,A0,A1,CV,DTM,
     *        D0,D1,Q,TM,TOLA
      DIMENSION
     *        CV(6,N)
      COMMON /MF1/
     *        Q,DUMMY(7)
```

```
      COMMON /MF2/
     *         A0,A,A1

      KA = 0

      IF (K.EQ.1) GO TO 20

      IF (APPROX-CV(1,K-1)) 10,10,20

   10 A0 = CV(1,K-1)+1.D0
                                            GO TO 30
   20 A0 = APPROX
   30 CALL TMOFA (A0,TM,DTM,M)

      IF (M.GT.0) GO TO 250

      D0 = -TM/DTM

      IF (D0) 100,300,50

C     A0 IS LOWER BOUND,
C     SEARCH FOR UPPER BOUND
   50 A1 = A0+D0+.1D0
      CALL TMOFA (A1,TM,DTM,M)

      IF (M.GT.0) GO TO 250

      D1 = -TM/DTM

      IF (D1) 200,350,60

   60 A0 = A1
      D0 = D1
      KA = KA+1

      IF (KA-4) 50,400,400

C     A1 IS UPPER BOUND,
C     SEARCH FOR LOWER BOUND
  100 A1 = A0
      D1 = D0
      A0 = DMAX1(A1+D1-.1D0,-2.D0*Q)

      IF (K.EQ.1) GO TO 110

      IF (A0-CV(1,K-1)) 150,150,110

  110 CALL TMOFA (A0,TM,DTM,M)

      IF (M.GT.0) GO TO 250

      D0 = -TM/DTM

      IF (D0) 120,300,200

  120 KA = KA+1

      IF (KA-4) 100,400,400

  150 KA = KA+1

      IF (KA-4) 160,400,400

  160 A0 = A1+DMAX1(TOLA,DABS(D1))
                                          GO TO 30
  200 A  = .5D0*(A0+D0+A1+D1)

      IF (A.LE.A0 .OR.
     *    A.GE.A1) A = .5D0*(A0+A1)

  250 MM = M
                                            RETURN
  300 CV(1,K) = A0
  310 CV(2,K) = 0.D0
      M       = -1
                                          GO TO 250
  350 CV(1,K) = A1
                                          GO TO 310
  400 M       = 2
                                          GO TO 250
      END
```

Algorithm 352 (Part A.2)

MFITR8 (Improves Characteristic Value)
(Called by MFCVAL)

Comments Given $a_0 < a < a_1$, where a_0 is a lower and a_1 an upper bound, the subroutine MFITR8 iterates to the characteristic value, replacing one of the bounds with a better approximation at each step. The process terminates after 40 iterations unless one of the following conditions occurs first: (1) $a - a_0 \leqq$ TOLA, (2) $a_1 - a \leqq$ TOLA, or (3) $|D(a)| <$ TOLA. See Appendix 3, method 2, of [3] for a detailed description of this process.

Explanation of output:

CV the characteristic value, a

DCV the function $D(a)$

MM an indicator cell used to communicate an error condition to subroutine MFCVAL.

MFITR8 calls on the subroutine:

TMOFA—referred to as Algorithm 352 (Part A.3)

```
      SUBROUTINE MFITR8 (TOLA,CV,DCV,MM)
C     ****************

      INTEGER
     *         M,MM,N

      DOUBLE PRECISION
     *         A,A0,A1,A2,CV,D,DCV,DTM,
     *         TM,TOLA

      LOGICAL
     *         LAST

      COMMON /MF2/
     *         A0,A,A1

      N    = 0
      LAST = .FALSE.
   50 N    = N+1
      CALL TMOFA (A,TM,DTM,M)

      IF (M.GT.0) GO TO 400

      D    = -TM/DTM

C     IS TOLERANCE MET
      IF (N       .EQ.  40 .OR.
     *    A-A0    .LE.TOLA .OR.
     *    A1-A    .LE.TOLA .OR.
     *    DABS(D).LT.TOLA) LAST = .TRUE.

      IF (D) 110,100,120

  100 CV   = A
      DCV  = 0.D0
                                          GO TO 320
C     REPLACE UPPER BOUND BY A
  110 A1   = A
                                          GO TO 200
C     REPLACE LOWER BOUND BY A
  120 A0   = A
  200 A2   = A+D

      IF (LAST) GO TO 300
      IF (A2.GT.A0.AND.A2.LT.A1)
     *                             GO TO 250

      A    = .5D0*(A0+A1)
                                          GO TO 50
  250 A    = A2
                                          GO TO 50

  300 IF (A2.LE.A0.OR.A2.GE.A1)
     *                             GO TO 350

      CALL TMOFA (A2,TM,DTM,M)

      IF (M.GT.0) GO TO 400

      D    = -TM/DTM
      CV   = A2
  310 DCV  = D
  320 MM   = M
                                            RETURN
  350 CV   = A
                                          GO TO 310
  400 CV   = 0.D0
      DCV  = 0.D0
                                          GO TO 320
      END
```

Algorithm 352 (Part A.3)

TMOFA (Accuracy Indicator)
(Called by MFCVAL, BOUNDS, MFITR8 and COEF)

Comments The subroutine TMOFA evaluates the function $T_m(a)$ and its derivative $dT_m(a)/da$. See [3] for the definitions, theorems, and numerical methods relating to the computation of these quantities.

Explanation of the arguments:

ALFA the given argument, a

TM $T_m(a)$

DTM $dT_m(a)/da$

ND internal error indicator cell

TMOFA calls no other subprograms.

```
      SUBROUTINE TMOFA (ALFA,TM,DTM,ND)
C     ****************

      INTEGER
     *         K,KK,KT,L,MF,M0,M1,M2S,
     *         ND,TYPE

      DOUBLE PRECISION
     *         A,AA,ALFA,B,DG,DTM,DTYPE,
     *         F,FL,G,H(200),HP,Q,QINV,
     *         Q1,Q2,T,TM,TOL,TT,V

      COMMON
     *         G(200,2),DG(200,2),AA,
     *         A(3),B(3),DTYPE,QINV,Q1,
     *         Q2,T,TT,K,L,KK,KT

      COMMON /MF1/
     *         Q,TOL,TYPE,M1,M0,M2S,MF

      EQUIVALENCE
     *         (H(1),G(1,1)),(Q1,HP),
     *         (Q2,F)

      DATA     FL /1.D+30/

C     STATEMENT FUNCTION
      V(K) = (AA-DBLE(FLOAT(K))**2)/Q

      ND       = 0
      KT       = 0
      AA       = ALFA
      DTYPE    = TYPE
      QINV     = 1.D0/Q
      DO 10        L = 1,2
         DO 5      K = 1,200
            G(K,L)   = 0.D0
            DG(K,L)  = 0.D0
    5    CONTINUE
   10 CONTINUE

      IF (MOD(TYPE,2)) 20,30,20

   20 M0       = 3
                                          GO TO 40
   30 M0       = TYPE+2
   40 K        = .5D0*DSQRT(DMAX1(
     *             3.D0*Q+AA,0.D0))
      M2S      = MIN0(2*K+M0+4,
     *                398+MOD(M0,2))

C     EVALUATION OF THE TAIL OF A
C     CONTINUED FRACTION
      A(1)     = 1.D0
      A(2)     = V(M2S+2)
      B(1)     = V(M2S)
      B(2)     = A(2)*B(1)-1.D0
      Q1       = A(2)/B(2)
      DO 50 K  = 1,200
         MF    = M2S+2+2*K
         T     = V(MF)
         A(3)  = T*A(2)-A(1)
         B(3)  = T*B(2)-B(1)
         Q2    = A(3)/B(3)

         IF (DABS(Q1-Q2).LT.TOL)
     *                           GO TO 70

         Q1    = Q2
         A(1)  = A(2)
         A(2)  = A(3)
         B(1)  = B(2)
         B(2)  = B(3)
```

```
   50 CONTINUE
      KT       = 1
   70 T        = 1.D0/T
      TT       = -T*T*QINV
      L        = MF-M2S
      DO 80 K  = 2,L,2
         T     = 1.D0/(V(MF-K)-T)
         TT    = T*T*(TT-QINV)
   80 CONTINUE
      KK       = M2S/2+1

      IF (KT.EQ.1) Q2 = T

      G(KK,2) = .5D0*(Q2+T)
      DG(KK,2)= TT

C     STAGE 1
      G(2,1)   = 1.D0
      DO 140   K = M0,M2S,2
         KK      = K/2+1

         IF (K.LT.5)
     *       IF (K-3) 100,110,120

         G(KK,1) = V(K-2)-1.D0/G(KK-1,1)
         DG(KK,1)= QINV+DG(KK-1,1)/
     *             G(KK-1,1)**2
                                      GO TO 130
  100    G(2,1)  = V(0)
         DG(2,1) = QINV
                                      GO TO 130
  110    G(2,1)  = V(1)+DTYPE-2.D0
         DG(2,1) = QINV
                                      GO TO 130
  120    G(3,1)  = V(2)+(DTYPE-2.D0)/
     *             G(2,1)
         DG(3,1) = QINV+(2.D0-DTYPE)*
     *             DG(2,1)/G(2,1)**2

         IF (TYPE.EQ.2) G(2,1) = 0.D0

  130    IF (DABS(G(KK,1)).LT.1.D0)
     *                         GO TO 200

  140 CONTINUE

C     BACKTRACK
      TM       = G(KK,2)-G(KK,1)
      DTM      = DG(KK,2)-DG(KK,1)
      M1       = M2S
      KT       = M2S-M0
      DO 180   L = 2,KT,2
         K       = M2S-L
         KK      = K/2+1
         G(KK,2) = 1.D0/(V(K)-G(KK+1,2))
         DG(KK,2)= -G(KK,2)**2*
     *             (QINV-DG(KK+1,2))

         IF (K-2) 150,150,160

  150    G(2,2)  = 2.D0*G(2,2)
         DG(2,2) = 2.D0*DG(2,2)
  160    T       = G(KK,2)-G(KK,1)

         IF (DABS(T)-DABS(TM))
     *                      170,180,180

  170    TM      = T
         DTM     = DG(KK,2)-DG(KK,1)
         M1      = K
  180 CONTINUE
                                      GO TO 320

C     STAGE 2
  200 M1       = K
      K        = M2S
      KK       = K/2+1

  210 IF (K.EQ.M1)
     *   IF (K-2) 300,300,310

      K        = K-2
      KK       = KK-1
      T        = V(K)-G(KK+1,2)

      IF (DABS(T)-1.D0) 250,220,220

  220 G(KK,2) = 1.D0/T
      DG(KK,2)= (DG(KK+1,2)-QINV)/T**2
                                      GO TO 210
C     STAGE 3
  250 IF (K.EQ.M1) IF (T) 220,290,220

      HP       = DG(KK+1,2)-QINV
  260 G(KK,2) = FL
      H(KK)    = T
      K        = K-2
      KK       = KK-1
      F        = V(K)*T-1.D0

      IF (K.EQ.M1) IF (F) 280,290,280

      IF (DABS(F)-DABS(T)) 270,280,280

  270 HP       = HP/T**2-QINV
      T        = F/T
                                      GO TO 260
  280 G(KK,2) = T/F
      DG(KK,2)= (HP-QINV*T*T)/F**2
                                      GO TO 210
  290 ND       = 1
                                      GO TO 320

C     CHAINING M EQUALS 2
  300 G(2,2)   = 2.D0*G(2,2)
      DG(2,2)  = 2.D0*DG(2,2)
  310 TM       = G(KK,2)-G(KK,1)
      DTM      = DG(KK,2)-DG(KK,1)
  320                                 RETURN
      END
```

Algorithm 352 (Part B.1)
COEF (Coefficients)
(Called by MATH)

Comments The subroutine COEF computes the *neutral* coefficients, as defined in the *Comments* of Algorithm 352 (Part B), and returns them via common array AB. Argument M is an internal error indicator cell. For details of the method used, see Appendix 6 of [3]. COEF calls on the subroutine: TMOFA—referred to as Algorithm 352 (Part A.3)

```
      SUBROUTINE COEF (M)
C     ***************

      INTEGER
     *        K,KA,KB,KK,M,MF,ML,MM,
     *        M0,M1,M2S,TYPE

      DOUBLE PRECISION
     *        A,AB,FL,G,H(200),Q,T,
     *        TOL,V,V2

      COMMON
     *        G(200,2),DUM1(800),A,T,K,
     *        KA,KB,KK,MM,ML,AB(200)

      COMMON /MF1/
     *        Q,TOL,TYPE,M1,M0,M2S,MF

      EQUIVALENCE
     *        (H(1),G(1,1))

      DATA    FL,V2/1.D+30,1.D-15/

C     STATEMENT FUNCTION
      V(K) = (A-DBLE(FLOAT(K))**2)/Q

      CALL TMOFA (A,T,T,M)

      IF (M.NE.0) GO TO 300

      DO 60    K = 1,200
         AB(K)   = 0.D0
   60 CONTINUE
      KA       = M1-M0+2
      DO 90    K = 2,KA,2
         KK      = (M1-K)/2+1

         IF (K-2) 70,70,80

   70    AB(KK)  = 1.D0
                                      GO TO 90
   80    AB(KK)  = AB(KK+1)/G(KK+1,1)
   90 CONTINUE
      KA       = 0
      DO 130   K = M1,M2S,2
         KK      = K/2+1
         ML      = K

         IF (G(KK,2).EQ.FL) GO TO 100

         AB(KK)  = AB(KK-1)*G(KK,2)
                                      GO TO 110
  100    T       = AB(KK-2)

         IF (K.EQ.4.AND.M1.EQ.2) T = T+T

         AB(KK)  = T/(V(K-2)*H(KK)-1.D0)

  110    IF (DABS(AB(KK)).GE.1.D-17)
     *                       KA = 0

         IF (KA.EQ.5) GO TO 260

         KA      = KA+1
  130 CONTINUE
      T        = DLOG(DABS(AB(KK))/V2)/
     *           DLOG(1.D0/DABS(G(KK,2)))
      KA       = 2*IDINT(T)
      ML       = KA+2+M2S

      IF (ML.GT.399) GO TO 400

      KB       = KA+2+MF
      T        = 1.D0/V(KB)
      KK       = MF-M2S
      DO 150   K = 2,KK,2
         T       = 1.D0/(V(KB-K)-T)
  150 CONTINUE
      KK       = ML/2+1
      G(KK,2) = T
      DO 200   K = 2,KA,2
         KK      = (ML-K)/2+1
         G(KK,2) = 1.D0/(V(ML-K)-
     *             G(KK+1,2))
  200 CONTINUE
      KA       = M2S+2
      DO 250   K = KA,ML,2
         KK      = K/2+1
         AB(KK)  = AB(KK-1)*G(KK,2)
  250 CONTINUE

C     NEUTRAL NORMALIZATION
  260 T        = AB(1)
      MM       = MOD(TYPE,2)
      KA       = MM+2
      DO 280   K = KA,ML,2
         KK      = K/2+1

         IF (DABS(T)-DABS(AB(KK)))
     *                      270,280,280

  270    T       = AB(KK)
         MM      = K
  280 CONTINUE
      DO 290   K = 1,KK
         AB(K)   = AB(K)/T
  290 CONTINUE
  300                                 RETURN
  400 M        = -1
                                      GO TO 300
      END
```

Algorithm 352 (Part B.2)
SUM (Series Evaluation)
(Called by MATH)

Comments The subroutine SUM performs the summation, truncating the series when the magnitude of two successive terms, relative to the magnitude of the largest term, is less than or equal to 10^{-13}.

If the user is willing to accept reduced accuracy, he may save some computing time by making this tolerance larger. On the other hand, however, a smaller tolerance will not necessarily increase the accuracy, since on a machine using 16-digit arithmetic the sum will be, at best, good to 16 digits.

The particular series being evaluated is determined by the arguments SOL and FNC within subroutine MATH and communicated to this subroutine via argument DUM.

Output is returned via common: variables F, DMAX, DLAST, KMAX, and KLAST.

SUM calls on one of the functions of Algorithm 352 (Part B.2.1).

```
      SUBROUTINE SUM (DUM)
C     **************

      INTEGER
     *          K,KLAST,KMAX,L,S

      DOUBLE PRECISION
     *          DLAST,DMAX,DUM,F,T

      COMMON
     *          DUM1(1006),S,L,DUM2(6),F,
     *          DMAX,DLAST,KMAX,KLAST,T

      K     = 0
      F     = DUM(0)
      DMAX  = F
      T     = DABS(F)
      KMAX  = 0
      DO 30 KLAST = 1,L
         DLAST = DUM(KLAST)
         F     = F+DLAST

         IF (T-DABS(DLAST)) 10,10,20

   10    DMAX  = DLAST
         T     = DABS(DMAX)
         KMAX  = KLAST

   20    IF (KLAST.LE.S) GO TO 30

         IF (DABS(DLAST)/T.GT.1.D-13)
     *                    K = 0

         K     = K+1

         IF (K.EQ.3) GO TO 40

   30 CONTINUE
      KLAST = L
   40                                   RETURN
      END
```

Algorithm 352 (Part C)
BESSEL (Bessel Functions)[2]
(Called by MATH)

Comments The subroutine BESSEL evaluates Bessel functions of the first or second kind, according as the argument SOL = 1 or 2, of orders $0, 1, \cdots, n$ and argument u, both of which must be nonnegative. Functions of order zero and one are always evaluated, regardless of the value of n. Results are returned via array JY, with element JY(K) containing the function of order K-1.

It should be noted that for SOL = 2 and $u = 0$, a large negative constant (-10^{37}) is returned as the function value for all orders and no warning is given.

Different methods of computation are used for $J_0(u)$, $J_1(u)$, $Y_0(u)$, and $Y_1(u)$, depending upon whether $u < 8$, or not. (See subroutines J0J1, Y0Y1, and LUKE for details.) The $J_n(u)$, $n = 2, 3, \cdots, m$, are computed by means of a continued fraction (see subroutine JNS), whereas the $Y_n(u)$ for corresponding orders are calculated directly from the recurrence relation:

$$Y_{n+1}(u) = \frac{2n}{u} Y_n(u) - Y_{n-1}(u)$$

BESSEL calls on the subroutines:
J0J1—referred to as Algorithm 352 (Part C.1)
Y0Y1—referred to as Algorithm 352 (Part C.2)
LUKE—referred to as Algorithm 352 (Part C.3)
JNS—referred to as Algorithm 352 (Part C.4)

```
      SUBROUTINE BESSEL (SOL,U,JY,N)
C     *****************

      INTEGER
     *          N,NN,SOL

      DOUBLE PRECISION
     *          JY(250),U

      NN = MIN0(N,249)

      IF (U.EQ.0.D0.AND.SOL.EQ.2)
     *                    GO TO 80

      IF (U.GE.8.D0) GO TO 30

      GO TO (10,20), SOL
   10 CALL J0J1 (U,JY)
                                      GO TO 40
   20 CALL Y0Y1 (U,JY)
                                      GO TO 40
   30 CALL LUKE (U,SOL,JY)

   40 IF (N.LT.2) GO TO 100

      GO TO (50,60), SOL
   50 CALL JNS (JY,U,NN)
                                      GO TO 100

C     RECURRENCE FORMULA
   60 DO 70     K = 2,NN
         JY(K+1) = 2.D0*
     *             DBLE(FLOAT(K-1))*
     *             JY(K)/U-JY(K-1)
   70 CONTINUE
                                      GO TO 100
   80 NN = NN+1
      DO 90  K = 1,NN
         JY(K) = -1.D+37
   90 CONTINUE
  100                                 RETURN
      END
```

Algorithm 352 (Part C.1)
J0J1 (First Kind)
(Called by BESSEL)

Comments The subroutine J0J1 computes the Bessel functions of the first kind, $J_0(x)$ and $J_1(x)$, for $x < 8$. This is done by evaluating formula 9.1.10 of [1]. The results are returned via array J.

J0J1 calls no other subprograms.

```
      SUBROUTINE J0J1 (X,J)
C     ***************

      DOUBLE PRECISION
     *          J(2),T(5),X

      COMMON
     *          DUM(1014),T

      T(1) = X/2.D0
      J(1) = 1.D0
      J(2) = T(1)
      T(2) = -T(1)**2
      T(3) = 1.D0
      T(4) = 1.D0

   10 T(4) = T(4)*T(2)/T(3)**2
      J(1) = J(1)+T(4)
      T(5) = T(4)*T(1)/(T(3)+1.D0)
      J(2) = J(2)+T(5)

      IF (DMAX1(DABS(T(4)),DABS(T(5)))
     *    .LT.1.D-15) RETURN

      T(3) = T(3)+1.D0
                                      GO TO 10
      END
```

Algorithm 352 (Part C.2)
Y0Y1 (Second Kind)
(Called by BESSEL)

Comments The subroutine Y0Y1 computes the Bessel functions of the second kind, $Y_0(x)$ and $Y_1(x)$, for $x < 8$. This is done by evaluating formulas 9.1.13 and 9.1.11 of [1]. The results are returned via array Y.

Y0Y1 calls no other subprograms.

```
      SUBROUTINE Y0Y1 (X,Y)
C     ***************

      DOUBLE PRECISION
     *          T(10),X,Y(2)

      COMMON
     *          DUM(1014),T

      T(1) = X/2.D0
      T(2) = -T(1)**2
      Y(1) = 1.D0
      Y(2) = T(1)
      T(7) = 0.D0
      T(10)= -T(1)
      T(3) = 0.D0
      T(4) = 0.D0
      T(5) = 1.D0

   10 T(3) = T(3)+1.D0
      T(4) = T(4)+1.D0/T(3)
      T(5) = T(5)*T(2)/T(3)**2
      Y(1) = Y(1)+T(5)
      T(6) = -T(5)*T(4)
      T(7) = T(7)+T(6)
      T(8) = T(5)*T(1)/(T(3)+1.D0)
      Y(2) = Y(2)+T(8)
      T(9) = -T(8)*(2.D0*T(4)+
     *       1.D0/(T(3)+1.D0))
      T(10)= T(10)+T(9)

      IF (DMAX1(DABS(T(6)),DABS(T(9)))
     *    .GE.1.D-15) GO TO 10

      T(2) = .57721566490153286D0+
     *       DLOG(T(1))
      Y(1) = .63661977236758134D0*
     *       (Y(1)*T(2)+T(7))
      Y(2) = .63661977236758134D0*
     *       (Y(2)*T(2)-1.D0/X)+T(10)/
     *       3.1415926535897932D0
                                      RETURN
      END
```

Algorithm 352 (Part C.3)
LUKE
(Called by BESSEL)

Comments The subroutine LUKE evaluates Bessel functions of order zero and one, of the first or second kind, according as the argument KIND = 1 or 2, for $u \geqq 8$. The results are returned via the 2-element array JY.

The Bessel function of the third kind (Hankel function), $H_\nu^{(1)}(u) = J_\nu(u) + iY_\nu(u)$, can be expressed in terms of the

[2] This subroutine (together with its subsidiary routines) may be removed in toto, with no changes, and used independently as a Bessel function algorithm. The results are good to 14 significant digits or decimal places, whichever is least accurate, with an error of no more than one unit in the last digit or place.

Chebyshev polynomials, $T_n^*(x)$, as follows:

$$H_\nu^{(1)}(u) = \left(\frac{2}{\pi u}\right)^{\frac{1}{2}} e^{i\left(u - \frac{\nu\pi}{2} - \frac{\pi}{4}\right)} \cdot \sum_{k=0}^{\infty} (\alpha_k^{(\nu)} + i\beta_k^{(\nu)}) T_k^*(R/u). \tag{9}$$

We now define $\alpha_k^{(0)} = A_{k+1}$, $\beta_\nu^{(0)} = B_{k+1}$, $\alpha_k^{(1)} = C_{k+1}$, $\beta_k^{(1)} = D_{k+1}$, $x = R/u$, and $T_k^*(x) = G_{k+1}(x)$. The recurrence relations for the $G_k(x)$ are as follows:

$$G_1(x) = 1, \qquad G_2(x) = 2x - 1,$$

$$G_k(x) = (4x-2)\, G_{k-1}(x) - G_{k-2}(x), \qquad k \geqq 3.$$

If we let $\nu = 0$ and make other appropriate substitutions in (9), while remembering that $e^{i\theta} = \cos\theta + i \sin\theta$, we can separate the real and imaginary parts and get the following relations:

$$J_0(u) = \left(\frac{2}{\pi u}\right)^{\frac{1}{2}} \cdot \left[\cos\theta \sum_{k=1}^{\infty} A_k\, G_k(x) - \sin\theta \sum_{k=1}^{\infty} B_k\, G_k(x)\right],$$

$$Y_0(u) = \left(\frac{2}{\pi u}\right)^{\frac{1}{2}} \cdot \left[\cos\theta \sum_{k=1}^{\infty} B_k\, G_k(x) + \sin\theta \sum_{k=1}^{\infty} A_k\, G_k(x)\right],$$

where $\theta = u - \pi/4$.

Notice that if $\nu = 1$ in (9), then θ is replaced by $\theta - \pi/2$. Also, $\cos(\theta - \pi/2) = \sin\theta$ and $\sin(\theta - \pi/2) = -\cos\theta$. Therefore, proceeding as before, we get

$$J_1(u) = \left(\frac{2}{\pi u}\right)^{\frac{1}{2}} \cdot \left[\sin\theta \sum_{k=1}^{\infty} C_k\, G_k(x) + \cos\theta \sum_{k=1}^{\infty} D_k\, G_k(x)\right],$$

$$Y_1(u) = \left(\frac{2}{\pi u}\right)^{\frac{1}{2}} \cdot \left[\sin\theta \sum_{k=1}^{\infty} D_k\, G_k(x) - \cos\theta \sum_{k=1}^{\infty} C_k\, G_k(x)\right].$$

The coefficients A, B, C, and D have been computed for $R = 8$ in eq. (9) and are guaranteed to the number of digits given.

LUKE calls no other subprograms.

```
      SUBROUTINE LUKE (U,KIND,JY)
C     ***************

      INTEGER
     *         K,KIND

      DOUBLE PRECISION
     *         A(19),B(19),CS,C(19),
     *         D(19),G(3),JY(2),R(2),
     *         S(2),SN,T,U,X

      COMMON
     *         DUM(1014),R,S,G,X,T,SN,CS

C     WARNING - THE FOLLOWING DATA
C     STATEMENTS ARE NOT IN ASA
C     STANDARD FORTRAN

      DATA A /
     *         .99959506476867287416D0,
     *        -.53807956139606913D-3,
     *        -.13179677123361570D-3,
     *         .151422497048644D-5,
     *         .15846861792063D-6,
     *        -.856069553946D-8,
     *        -.29572343355D-9,
     *         .6573556254D-10,
     *        -.223749703D-11,
     *        -.44821140D-12,
     *         .6954827D-13,
     *        -.151340D-14,
     *        -.92422D-15,
     *         .15558D-15,
     *        -.476D-17,
     *        -.274D-17,
     *         .61D-18,
     *        -.4D-19,
     *        -.1D-19/

      DATA B /
     *        -.77693556942053213 6D-2,
     *        -.77480323096544767 0D-2,
     *         .25365411654307960-4,
     *         .394273598399711D-5,
     *        -.107234982991 29D-6,
     *        -.721389799328D-8,
     *         .73764602893D-9,
     *         .150687811D-11,
     *        -.574589537D-11,
     *         .45996574D-12,
     *         .2270323D-13,
     *        -.887890D-14,
     *         .74497D-15,
     *         .5847D-16,
     *        -.2410D-16,
     *         .265D-17,
     *         .13D-18,
     *        -.10D-18,
     *         .2D-19/

      DATA C /
     *        1.00067753586591346234D0,
     *         .90100725195908183D-3,
     *         .22172434918599454D-3,
     *        -.196575946319104D-5,
     *        -.20889531143270D-6,
     *         .10281443508 94D-7,
     *         .37597054789D-9,
     *        -.7638891358D-10,
     *         .238734670D-11,
     *         .51825489D-12,
     *        -.7693969D-13,
     *         .144008D-14,
     *         .103294D-14,
     *        -.16821D-15,
     *         .459D-17,
     *         .302D-17,
     *        -.65D-18,
     *         .4D-19,
     *         .1D-19/

      DATA D /
     *         .23376829986285803 28D-1,
     *         .23346801223545575 33D-1,
     *        -.35760105909013 82D-4,
     *        -.56086314949262 7D-5,
     *         .13273894084340D-6,
     *         .916975845066D-8,
     *        -.86838880371D-9,
     *        -.378073005D-11,
     *         .663145586D-11,
     *        -.50584390D-12,
     *        -.2720782D-13,
     *         .985381D-14,
     *        -.79398D-15,
     *        -.6757D-16,
     *         .2625D-16,
     *        -.280D-17,
     *        -.15D-18,
     *         .10D-18,
     *        -.2D-19/

      X    = 8.D0/U
      G(1) = 1.D0
      G(2) = 2.D0*X-1.D0
      R(1) = A(1)+A(2)*G(2)
      S(1) = B(1)+B(2)*G(2)
      R(2) = C(1)+C(2)*G(2)
      S(2) = D(1)+D(2)*G(2)
      DO 10 K = 3,19
         G(3) = (4.D0*X-2.D0)*G(2)-G(1)
         R(1) = R(1)+A(K)*G(3)
         S(1) = S(1)+B(K)*G(3)
         R(2) = R(2)+C(K)*G(3)
         S(2) = S(2)+D(K)*G(3)
         G(1) = G(2)
         G(2) = G(3)
   10 CONTINUE

      T  = .7978845608028654D0/DSQRT(U)
      SN = DSIN(U-.7853981633974483D0)
      CS = DCOS(U-.7853981633974483D0)

      GO TO (20,30), KIND
   20 JY(1) = T*(R(1)*CS-S(1)*SN)
      JY(2) = T*(R(2)*SN+S(2)*CS)
                                GO TO 40
   30 JY(1) = T*(S(1)*CS+R(1)*SN)
      JY(2) = T*(S(2)*SN-R(2)*CS)
   40                                  RETURN
      END
```

Algorithm 352 (Part C.4)

JNS

(Called by BESSEL)

Comments The subroutine JNS evaluates Bessel functions of the first kind, of orders $n = 2, 3, \cdots, m$, for argument u, given $J_0(u)$ and $J_1(u)$. From the definition $G_n = J_n(u)/J_{n-1}(u)$ and the recurrence relation,

$$J_{n+1}(u) = (2n/u)\; J_n(u) - J_{n-1}(u),$$

we can derive the following equation:

$$G_n = \frac{1}{\dfrac{2n}{u} - G_{n+1}}. \tag{10}$$

Since G_{n+1} is of the same form as G_n, we can continue the process and obtain the continued fraction,

$$G_n = \frac{1}{\dfrac{2n}{u}} - \frac{1}{\dfrac{2(n+1)}{u}} - \cdots - \frac{1}{\dfrac{2(n+k)}{u} - G_{n+k+1}}. \tag{11}$$

G_m is evaluated using (11), then the other G_n are computed from (10) for $n = m - 1, m-2, \cdots, 2$. Finally, the J_n are evaluated in a forward direction from $J_n = G_n J_{n-1}$ and returned via argument array JJ. See [2] for a more detailed treatment of this process.

JNS calls no other subprograms.

```
      SUBROUTINE JNS (JJ,U,M)
C     **************

      INTEGER
     *         K,KA,KK,M

      DOUBLE PRECISION
     *         A,B,D(2),DM,G(249),
     *         JJ(250),P(3),Q(3),U

      EQUIVALENCE
     *         (A,G),(D,G(2)),
     *         (P,G(4)),(Q,G(7)),
     *         (DM,G(10)),(B,G(11))

      COMMON
     *         DUM(1014),G,M,K,KK,KA

      DM   = 2*M
      P(1) = 0.D0
      Q(1) = 1.D0
      P(2) = 1.D0
      Q(2) = DM/U
      D(1) = P(2)/Q(2)
      A    = 2.D0
```

```
10 B     = (DM+A)/U
   P(3) = B*P(2)-P(1)
   Q(3) = B*Q(2)-Q(1)
   D(2) = P(3)/Q(3)

   IF (DABS(D(1)-D(2))
  *    .LT.1.D-15) GO TO 20

   P(1) = P(2)
   P(2) = P(3)
   Q(1) = Q(2)
   Q(2) = Q(3)
   D(1) = D(2)
   A    = A+2.D0
                              GO TO 10
20 G(M) = D(2)
   KA   = M-2
   DO 30  K = 1,KA
      KK    = M-K
      A     = 2*KK
      G(KK) = U/(A-U*G(KK+1))

      IF (G(KK).EQ.0.D0)
  *              G(KK) = 1.D-35

30 CONTINUE
   DO 40    K = 2,M
      JJ(K+1) = G(K)*JJ(K)
40 CONTINUE
                               RETURN
   END
```

Algorithm 352 (Part B.2.1)
DS, DC, DDS, DDC, PS, PC, DPS, DPC
(Called by MATH via SUM)

Comments The following collection of function subprograms is utilized by SUM to evaluate the kth term ($k = 0, 1, \cdots$) of one of the following: eq. (2), (3), (5), (6), or their derivatives.

DS and DC call on functions FJ and FY.

DDS and DDC call on functions FJ, FY, DJ and DY.

PS, PC, DPS, and DPC call no other subprograms.

```
      DOUBLE PRECISION FUNCTION DS(KK)
C     ************************

      INTEGER
     *        K,KK,N,N1,N2,P,S

      DOUBLE PRECISION
     *        AB,FJ,FY

      COMMON
     *        DUM1(1004),N,P,S,DUM2(17),
     *        K,N1,N2,DUM3(583),AB(200)

C     EVALUATES ONE TERM OF THE RADIAL
C     SOLUTION, ASSOCIATED WITH B(Q)
      K  = KK
      N1 = K-S
      N2 = K+S+P
      DS = AB(K+1)*(FJ(N1)*FY(N2)-
     *               FJ(N2)*FY(N1))

      IF (MOD(K+N,2).NE.0) DS = -DS

                                RETURN
      END


      DOUBLE PRECISION FUNCTION DC(KK)
C     ************************

      INTEGER
     *        K,KK,N,N1,N2,P,S

      DOUBLE PRECISION
     *        AB,FJ,FY

      COMMON
     *        DUM1(1004),N,P,S,DUM2(17),
     *        K,N1,N2,DUM3(583),AB(200)

C     EVALUATES ONE TERM OF THE RADIAL
C     SOLUTION, ASSOCIATED WITH A(Q)
      K  = KK
      N1 = K-S
      N2 = K+S+P
      DC = AB(K+1)*(FJ(N1)*FY(N2)+
     *               FJ(N2)*FY(N1))

      IF (MOD(K+N,2).NE.0) DC = -DC

      IF (S+P.EQ.0) DC = .5D0*DC

                                RETURN
      END


      DOUBLE PRECISION FUNCTION DDS(KK)
C     ************************

      INTEGER
     *        K,KK,N,N1,N2,P,S

      DOUBLE PRECISION
     *        AB,DJ,DY,FJ,FY,U1,U2

      COMMON
     *        DUM1(1000),U1,U2,N,P,S,
     *        DUM2(17),K,N1,N2,
     *        DUM3(583),AB(200)

C     EVALUATES ONE TERM OF THE DERIVATIVE
C     OF THE RADIAL SOLUTION,
C     ASSOCIATED WITH B(Q)
      K   = KK
      N1  = K-S
      N2  = K+S+P
      DDS = AB(K+1)*(U2*(FJ(N1)*DY(N2)-
     *      FJ(N2)*DY(N1))-U1*(FY(N2)*
     *      DJ(N1)-FY(N1)*DJ(N2)))

      IF (MOD(K+N,2).NE.0) DDS = -DDS

                                RETURN
      END


      DOUBLE PRECISION FUNCTION DDC(KK)
C     ************************

      INTEGER
     *        K,KK,N,N1,N2,P,S

      DOUBLE PRECISION
     *        AB,DJ,DY,FJ,FY,U1,U2

      COMMON
     *        DUM1(1000),U1,U2,N,P,S,
     *        DUM2(17),K,N1,N2,
     *        DUM3(583),AB(200)

C     EVALUATES ONE TERM OF THE DERIVATIVE
C     OF THE RADIAL SOLUTION,
C     ASSOCIATED WITH A(Q)
      K   = KK
      N1  = K-S
      N2  = K+S+P
      DDC = AB(K+1)*(U2*(FJ(N1)*DY(N2)+
     *      FJ(N2)*DY(N1))-U1*(FY(N2)*
     *      DJ(N1)+FY(N1)*DJ(N2)))

      IF (MOD(K+N,2).NE.0) DDC = -DDC

      IF (S+P.EQ.0) DDC = .5D0*DDC

                                RETURN
      END


      DOUBLE PRECISION FUNCTION PS(K)
C     ************************

      INTEGER
     *        K,P

      DOUBLE PRECISION
     *        AB,X

      COMMON
     *        DUM1(1005),P,DUM2(2),X,
     *        DUM3(600),AB(200)

C     EVALUATES ONE TERM OF THE ODD
C     PERIODIC SOLUTION
      PS = AB(K+1)*
     *     DSIN(DBLE(FLOAT(2*K+P))*X)
                                RETURN
      END


      DOUBLE PRECISION FUNCTION PC(K)
C     ************************

      INTEGER
     *        K,P

      DOUBLE PRECISION
     *        AB,X

      COMMON
     *        DUM1(1005),P,DUM2(2),X,
     *        DUM3(600),AB(200)

C     EVALUATES ONE TERM OF THE EVEN
C     PERIODIC SOLUTION
      PC = AB(K+1)*
     *     DCOS(DBLE(FLOAT(2*K+P))*X)
                                RETURN
      END


      DOUBLE PRECISION FUNCTION DPS(K)
C     ************************

      INTEGER
     *        K,P

      DOUBLE PRECISION
     *        AB,T,X

      COMMON
     *        DUM1(1005),P,DUM2(2),X,
     *        DUM3(14),T,DUM4(584),
     *        AB(200)

C     EVALUATES ONE TERM OF THE DERIVATIVE
C     OF THE ODD PERIODIC SOLUTION
      T   = 2*K+P
      DPS = AB(K+1)*T*DCOS(T*X)
                                RETURN
      END


      DOUBLE PRECISION FUNCTION DPC(K)
C     ************************

      INTEGER
     *        K,P

      DOUBLE PRECISION
     *        AB,T,X

      COMMON
     *        DUM1(1005),P,DUM2(2),X,
     *        DUM3(14),T,DUM4(584),
     *        AB(200)

C     EVALUATES ONE TERM OF THE DERIVATIVE
C     OF THE EVEN PERIODIC SOLUTION
      T   = 2*K+P
      DPC = -AB(K+1)*T*DSIN(T*X)
                                RETURN
      END
```

Algorithm 352 (Part B.2.2)
FJ, FY, DJ, DY (Bessel Functions and Derivatives)
(Called by DS, DC, DDS, DDC)

Comments The following collection of function subprograms produces Bessel functions or their derivatives for integer order n, n being positive or negative. This is accomplished by using the already computed functions of nonnegative order (Algorithm 352 (Part C)) and substituting them in one of the following formulas:

$$J_{-n}(u) = (-1)^n J_n(u),$$

$$Y_{-n}(u) = (-1)^n Y_n(u),$$

$$J_n'(u) = \frac{n}{u} J_n(u) - J_{n+1}(u),$$

$$Y_n'(u) = Y_{n-1}(u) - \frac{n}{u} Y_n(u),$$

whichever is appropriate.
DJ calls on function FJ.
DY calls on function FY.
FJ and FY call no other subprograms.

```
      DOUBLE PRECISION FUNCTION FJ(N)
C     ************************
      INTEGER
     *          K,N
      DOUBLE PRECISION
     *          J
      COMMON
     *          J(250),DUM(527),K
C   PRODUCES BESSEL FUNCTIONS
C   OF THE FIRST KIND
      K  = IABS(N)
      IF (K.GE.250) GO TO 20
      FJ = J(K+1)
      IF (MOD(N,2).LT.0) FJ = -FJ
   10                                   RETURN
   20 FJ = 0.D0
      WRITE (6,99) N
   99 FORMAT(2H0J13,7H NEEDED)
                                        GO TO 10
      END

      DOUBLE PRECISION FUNCTION FY(N)
C     ************************
      INTEGER
     *          K,N
      DOUBLE PRECISION
     *          Y
      COMMON
     *          DUM1(500),Y(250),DUM2(27),K
C   PRODUCES BESSEL FUNCTIONS
C   OF THE SECOND KIND
      K  = IABS(N)
      IF (K.GE.250) GO TO 20
      FY = Y(K+1)
      IF (MOD(N,2).LT.0) FY = -FY
   10                                   RETURN
   20 FY = 0.D0
      WRITE (6,99) N
   99 FORMAT(2H0YI3,7H NEEDED)
                                        GO TO 10
      END

      DOUBLE PRECISION FUNCTION DJ(N)
C     ************************
      INTEGER
     *          N
      DOUBLE PRECISION
     *          FJ,FN,U1
      COMMON
     *          DUM1(1000),U1,DUM2(26),FN
C   DERIVATIVES OF BESSEL FUNCTIONS
C   OF THE FIRST KIND
      FN = N
      IF (N-249) 10,20,40
   10 DJ = FN*FJ(N)/U1-FJ(N+1)
                                        GO TO 30
   20 DJ = FJ(N-1)-FN*FJ(N)/U1
   30                                   RETURN
   40 DJ = 0.D0
      WRITE (6,99) N
   99 FORMAT(3H0J@I3,7H NEEDED)
                                        GO TO 30
      END

      DOUBLE PRECISION FUNCTION DY(N)
C     ************************
      INTEGER
     *          N
      DOUBLE PRECISION
     *          FN,FY,U2
      COMMON
     *          DUM1(1002),U2,DUM2(24),FN
C   DERIVATIVES OF BESSEL FUNCTIONS
C   OF THE SECOND KIND
      IF (N.GE.250) GO TO 20
      FN = N
      DY = FY(N-1)-FN*FY(N)/U2
   10                                   RETURN
   20 DY = 0.D0
      WRITE (6,99) N
   99 FORMAT(3H0Y@I3,7H NEEDED)
                                        GO TO 10
      END
```

REMARK ON ALGORITHM 268 [R2]
ALGOL 60 REFERENCE LANGUAGE EDITOR

[W. M. McKeeman, *Comm. ACM 8* (Nov. 1965), 667]

G. Sauer (Recd. 23 Dec. 1968)
Institut für Theoretische Physik der Justus-Liebig-Universität, 63 Giessen, West Germany

KEY WORDS AND PHRASES: symbol manipulation
CR CATEGORIES: 4.49

In the **procedure** *send*, replace the line
1 until 1 **do if** *buffer*[u+1] =
with the line
1 until *tabstop* **do if** *buffer*[u+1] = (1)
The published version fails to clear the buffer when a line to be printed contains no blanks and *tabstop* > 0, causing an array bounds violation. Knowing *buffer*[*tabstop*+1] never to contain a blank character, the search for blanks may be stopped at u = *tabstop* + 1.

(1) The author is indebted to the referee for suggesting this brief form.

The policy concerning the contributions of algorithms to *Communications of the ACM* appears, most recently, in the January 1969 issue, page 39. A contribution should be in the form of an algorithm, a certification, or a remark. An algorithm must normally be written in the ALGOL 60 Reference Language or in USASI Standard FORTRAN or Basic FORTRAN.

REMARK ON ALGORITHM 274 [F1]
GENERATION OF HILBERT DERIVED TEST MATRIX

[J. Boothroyd, *Comm. ACM 9* (Jan. 1966), 11]

J. Boothroyd (Recd. 7 Jan. 1969)
University of Tasmania, Hobart, Tasmania, Australia

KEY WORDS AND PHRASES: test matrix, Hilbert matrix
CR CATEGORIES: 5.14

An alternative, simpler, and more efficient procedure for generating test matrices having the same properties as those generated by Algorithm 274 is given below. The method, like that of Algorithm 274, is due to T. J. Dekker and may be described as follows.

The elements of the inverse of a segment of a Hilbert matrix are given by

$$(H^{-1}) = (-1)^{i+j} \times f_i \times f_j/(i+j-1)$$

where

$$f_i = \mathit{factorial}\ (n+i-1)/(\mathit{factorial}\ (i-1)) \uparrow 2/\mathit{factorial}\ (n-i).$$

The f_i may be factored as $f_i = f_{i1} \times f_{i2}$, in which

$$f_{i1} = \binom{n+i-1}{i-1} \times n, \quad f_{i2} = \binom{n-1}{n-i}.$$

Test matrices T are constructed by $T = D_1 H D_2$ where $D_1 = \mathit{diag}\ (f_{i1})$, $D_2 = \mathit{diag}\ (f_{i2})$, and H is the Hilbert matrix segment $H_{i,j} = 1/(i+j-1)$. It may be seen that this is equivalent to defining the T matrices by:

$$T_{i,j} = (fi)(fj)/(i+j-1),$$

$$fi = \binom{n+i-1}{i-1} \times n, \quad fj = \binom{n-1}{n-j},$$

with fi, fj given by the recurrence relations:

$$(fi)_1 = n, \quad (fi)_{i+1} = (fi)_i \times (n + i)/i,$$

$$(fj)_1 = 1, \quad (fj)_{j+1} = (fj)_j \times (n - j)/j.$$

That the condition $K(T)$ of these matrices is severe may be seen from an observation of the referee, who notes that

$$K(T) = \| T \| \times \| T^{-1} \| ,$$

$$\geq \max_{1 \leq i,j \leq n} (t_{i,j})\uparrow 2 = (t_{n,(n+1)} \div 2)\uparrow 2 \sim (2\uparrow 3n/13n)\uparrow 2,$$

where $\| \cdot\cdot \|$ is the L_1, L_2, L_∞, or the Euclidean matrix norm.

Other properties of these matrices shared by those of Algorithm 274 are:

(a) Each matrix has unit determinant;

(b) The eigenvalues form a set $\lambda_1, \lambda_2, \cdots, 1/\lambda_2, 1/\lambda_1$, so that odd order matrices have one eigenvalue of unity.

The procedure *testmx*1 below has been tested on an Elliott 503 (positive integer word length of 38 bits) and matrices of all orders up to 13 were generated before integer overflow occurred with $n = 14$.

procedure *testmx*1 (*a*, *n*); **value** *n*; **integer** *n*; **array** *a*;
comment generates in $a[1 : n, 1 : n]$ test matrices with integer elements given by

$$t_{i,j} = \binom{n+i-1}{i-1} \times n \times \binom{n-1}{n-j} / (i + j - 1)$$

and such that the elements of T inverse are $(-1)^{i+j} \times t_{i,j}$.

To determine for a particular computer that limit on n which permits the exact machine representation of all elements of these matrices, the following maximum values are listed:

n	$t_{i,j}$ (max)
8	163800
9	1178100
10	8314020
11	61108047
12	440936496;

```
begin
  integer i, j, fi, fj, iless1;
  fi := n;  iless1 := 0;
  for i := 1 step 1 until n do
  begin
    fj := 1;
    for j := 1 step 1 until n do
    begin
      a[i, j] := (fi×fj) ÷ (iless1+j);
      fj := ((n−j)×fj) ÷ j
    end;
    fi := ((n+i)×fi) ÷ i;  iless1 := i
  end
end testmx1
```

Proofs that the test matrices described above have integer elements and checkerboard inverses follow the lines of similar proofs given in [1].

Acknowledgments: Thanks are due to T. J. Dekker for communicating details of this method and to the referee for the contribution mentioned.

Reference:

1. Dekker, T. J. Evaluation of determinants, solution of systems of linear equations and matrix inversion. Rep. No. MR63, Mathematical Centre, Amsterdam, June 1963, pp. 8 and 9.

REMARK ON ALGORITHM 333 [H] MINIT ALGORITHM FOR LINEAR PROGRAMMING [Rodolfo C. Salazar and Subrata K. Sen, *Comm. ACM 11* (June 1968), 437]

D. K. Messham (Recd. 27 Nov. 1968 and 28 Feb. 1969)
Nelson Research Laboratories, The English Electric Co. Ltd., Stafford, England

KEY WORDS AND PHRASES: linear programming, dual simplex method, primal problem, dual problem
CR CATEGORIES: 5.41

The procedure has been tested with Marconi Myriad Algol, and it ran successfully when the following changes had been made (the first is merely a misprint):

1. The first statement in procedure *results* was changed
from $z := e[1 : lcol]$;
to $z := e[1, lcol]$;

2. To satisfy an Algol 60 restriction that a type procedure should contain an assignment to its procedure identifier, the **real** on the first line of the procedure was removed.

3. It is possible for the published algorithm to give incorrect results when it reaches a state in *phase*1 where there are no possible pivotal elements in one column of the tableau. (For example, maximize $- x_1 - x_2 - x_3$, with $2x_1 + x_2 = 3$ and $x_3 = 1$, reaches this state.) To correct this the line in procedure *phase*1
if *gamma* < *gmin* **then**
was changed to
if *gamma* < *gmin* ∧ *thmin* [*ind*[*k*]] < ₁₀6 **then**
All the appearances of 10^6 in this algorithm should be written as ₁₀6.

The following improvements are also suggested:

4. It is assumed that *lcol* is a global integer with the correct value. This was made unnecessary by adding *lcol* to the list of integers declared on the line immediately following the initial comment; the bounds of the array *ind*, declared on the next line, were changed
from [1 : *lcol*]
to [1 : *m*+*n*−*p*+1];
and *lcol* := *m* + *n* − *p* + 1;
was inserted as the first executable statement of the procedure *MINIT* (after **end** *phase*1;).

5. It is assumed that equality constraints will be given with positive right-hand sides. This restriction was overcome by inserting in the procedure *phase*1 after the line **integer array** *imin* [1 : *lcol*]; the following:

```
for i := m − p + 2 step 1 until m + 1 do
if e[i, lcol] < 0 then
for j := 1 step 1 until lcol do e[i, j] := − e[i, j];
```