

Bessel Functions of Real Argument and Integer Order

David J. Sookne¹

Institute for Basic Standards, National Bureau of Standards, Washington, D.C. 20234

(June 4, 1973)

A computer program is described for calculating Bessel Functions $J_n(x)$ and $I_n(x)$, for x real, and n a nonnegative integer. The method used is that of backward recursion, with strict control of error.

Key words: Backward recursion; Bessel functions; difference equation; error bounds; Miller algorithms.

1. Method

Given a real number x and a positive integer NB , BESLRI calculates either

$$I_n(x), \quad n = 0, 1, \dots, NB - 1$$

or

$$J_n(x), \quad n = 0, 1, \dots, NB - 1$$

using double-precision arithmetic. The method, which is described in [1],² is based on algorithms of Olver [2] and Miller [3], applied to the difference equation

$$y_{n-1} = \frac{2n}{x} y_n - \text{SIGN} \cdot y_{n+1} \quad (1)$$

where SIGN is +1 for J 's, -1 for I 's.

The program sets $\text{MAGX} = [|x|]$, the integer part of $|x|$, $p_{\text{MAGX}} = 0$, $p_{\text{MAGX}+1} = 1$, and then successively calculates

$$p_{n+1} = \frac{2n}{|x|} p_n - \text{SIGN} \cdot p_{n-1} \quad n = \text{MAGX} + 1, \text{MAGX} + 2, \dots \quad (2)$$

The sequence is strictly increasing. The program takes N to be the least n such that p_n exceeds a number TEST defined in sections 2 and 3. It then sets $y_N^{(N)} = 0$, $y_{N-1}^{(N)} = 1/p_N$, and recurs backward using (1). The computed sequence $y_0^{(N)}, y_1^{(N)}, \dots$ is the recessive solution of (1) which satisfies the boundary condition $y_{\text{MAGX}} = 1$. From this solution, the I 's and J 's are found by normalizing

$$J_n(x) = y_n^{(N)}/\mu \quad n = 0, 1, \dots, NB - 1$$

$$I_n(x) = y_n^{(N)}/\mu \quad n = 0, 1, \dots, NB - 1$$

AMS Subject Classification: 68A10.

¹ Present address: W.U.J.S. Institute, Arad, Israel.

² Figures in brackets indicate the literature references at the end of this paper.

where

$$\mu = y_0^{(N)} + 2 \sum_{k=1}^{\lfloor N/2 \rfloor} y_{2k}^{(N)} \quad \text{for } J\text{'s}$$

$$\mu = \left(y_0^{(N)} + 2 \sum_{k=1}^{\lfloor N/2 \rfloor} y_{2k}^{(N)} \right) / \cosh x \quad \text{for } I\text{'s.} \quad (3)$$

2. Error Bounds for the y_n

For $n > \text{MAGX}$, the truncation error in $y_n^{(N)}$ is

$$T_n^{(N)} = y_n - y_n^{(N)} = p_n \sum_{r=N}^{\infty} \frac{1}{p_r p_{r+1}};$$

see [2], eqs (5.01) and (5.02). This error is bounded by using Lemma 2 of [1], which states that for $s \geq r$, $\rho_s = p_{s+1}/p_s \geq \min(p_{r+1}/p_r, (r+1)/|x| + \sqrt{(r+1)^2/|x|^2 - 1})$.

The program sets

$$\text{TEST}_1 \geq \sqrt{2 \cdot 10^{\text{NSIG}} p_L p_{L+1}}, \quad (4)$$

where $L = \max(\text{MAGX} + 1, \text{NB} - 1)$, and NSIG is the maximum number of significant decimal digits in a double-precision variable on the computer being used. Then N' is the least n such that $p_n > \text{TEST}_1$. For J 's, N is the least $n \geq N'$ such that

$$p_n > \text{TEST} = \sqrt{\frac{\rho_{N'}}{\rho_{N'}^2 - 1}} \cdot \text{TEST}_1. \quad (5)$$

In consequence of (4) and (5), the relative truncation error $|T_n^{(N)}/y_n|$ is less than $\frac{1}{2} \cdot 10^{-\text{NSIG}}$ for all n in the range $\text{MAGX} < n \leq L$; see [1, sec. 5]. For I 's, N is taken to be N' . Here a glance at (1) shows $p_n/p_{n-1} > 2$ for all $n > \text{MAGX}$, so the proof given in [1, sec. 5] for J 's may also be used for I 's, with ρ_N replaced by 2 throughout.

In eq (1) for I 's, $\frac{2n}{x} y_n^{(N)}$ and $-\text{SIGN} \cdot y_{n+1}^{(N)}$ have the same sign since if $x > 0$, then $y_n^{(N)} > 0$ for all n , and if $x < 0$, then consecutive $y_n^{(N)}$'s have opposite signs. Thus relative errors $< \epsilon$ in $y_{n+1}^{(N)}$ and $y_n^{(N)}$ will produce a relative error $< \epsilon$ in $y_{n-1}^{(N)}$. Therefore, the relative truncation error in all $I_n(x)$, $0 \leq x < \text{NB}$, is bounded by $\frac{1}{2} \cdot 10^{-\text{NSIG}}$.

For $J_n(x)$, $n \leq \text{MAGX}$, the relative truncation error cannot be bounded, since as $n = \text{MAGX}$, $\text{MAGX} - 1, \dots, 1$ in (1), the values $J_n(x)$ oscillate, and precision is lost owing to cancellation. In this range, $J_n(x)$ is accurate to about D decimal places, where D is the number of decimals in $J_{\text{MAGX}+1}(x)$ which corresponds to NSIG significant figures in the same quantity [1, sec. 5].

3. Normalization and Error Bounds for μ

The eqs (3) were chosen to keep cancellation under control. First,

$$|J_0(x)| \leq 1, \quad |J_n(x)| \leq \frac{1}{\sqrt{2}}, \quad \text{for } n \geq 1 \quad (6)$$

[4, 2.5], so each term of the sum (3) for J 's is less than $\sqrt{2}$ times the whole sum. Since (6) is a rather weak bound, cancellation is even less than this would indicate.

For I 's, cancellation is avoided altogether, since all terms in the sum (3) for I 's have the same sign, as shown in section 2.

Besides bounding the truncation error of the algorithm, the program provides an estimated bound for the truncation error of the normalization sum, defined by eqs (3). For J 's, this error is

$$S^{(N)} = y_0 - y_0^{(N)} + 2 \sum_{k=1}^{\infty} (y_{2k} - y_{2k}^{(N)}).$$

For $2k \leq \text{MAGX}$, a bound for the error term $y_{2k}^{(N)} - y_{2k}$ is unavailable. For $\text{MAGX} < 2k < N$, $|y_{2k}^{(N)} - y_{2k}| < p_{2k} \rho_N / (\rho_N^2 - 1)$ [1, sec. 5]. To avoid storing all the p_{2k} , the program allows only for terms for which $2k \geq N$. Here $y_{2k}^{(N)} = 0$, and $y_{2k} = p_{2k} \sum_{r=2k}^{\infty} 1/(p_r p_{r+1})$. Therefore

$$\begin{aligned} |y_{2k}^{(N)} - y_{2k}| &= \left| \frac{1}{p_{2k+1}} \left\{ 1 + \frac{p_{2k}}{p_{2k+2}} + \frac{p_{2k}}{p_{2k+2}} \frac{p_{2k+1}}{p_{2k+3}} + \dots \right\} \right| \\ &\leq \frac{1}{p_{2k+1}} \left\{ 1 + \frac{1}{\rho_{2k}^2} + \frac{1}{\rho_{2k}^4} + \dots \right\} = \frac{\rho_{N'}^2}{(\rho_{N'}^2 - 1)p_{2k+1}}; \end{aligned}$$

compare section 2 above. Now let

$$R^{(N)} = \sum_{k = \lceil \frac{N+1}{2} \rceil}^{\infty} |y_{2k}|.$$

Then

$$\begin{aligned} R^{(N)} &\leq \frac{\rho_{N'}^2}{\rho_{N'}^2 - 1} \sum_{k = \lceil \frac{N+1}{2} \rceil}^{\infty} \frac{1}{p_{2k+1}} \leq \frac{\rho_{N'}^2}{(\rho_{N'}^2 - 1)p_{N+1}} \left\{ 1 + \frac{1}{\rho_{N'}^2} + \frac{1}{\rho_{N'}^4} + \dots \right\} \\ &\leq \frac{\rho_{N'}^3}{(\rho_{N'}^2 - 1)^2 p_N}. \end{aligned} \quad (7)$$

For J 's the program sets $\text{TEST}_1 \geq 2 \cdot 10^{\text{NSIG}}$. The normalization factor μ is $1/J_{\text{MAGX}}(x)$ [1, sec. 5],

so

$$\left| \frac{R^{(N)}}{\mu} \right| = \left| J_{\text{MAGX}}(x) \right| R^{(N)} \leq \frac{|J_{\text{MAGX}}(x)| \rho_{N'}^3}{(\rho_{N'}^2 - 1)^2} \cdot \frac{1}{2} \cdot 10^{-\text{NSIG}}.$$

This is a rather weak upper bound, and the error $\left| \frac{S^{(N)}}{\mu} \right|$ turns out to be less than $\frac{1}{2} \cdot 10^{-\text{NSIG}}$.

For I 's, $p_n/p_{n-1} > 2$ so the analog of (7) is

$$R^{(N)} \leq \frac{8}{9p_N \cosh x};$$

this is derived by substituting 2 for ρ_N in (7), and introducing the factor $\cosh x$; see (3).

By setting $\text{TEST}_1 \geq \frac{2 \cdot 10^{-\text{NSIG}}}{\exp(0.461 \cdot \text{MAGX})}$, the program insures that

$$R^{(N)} \leq \frac{4 \exp(0.461 \cdot \text{MAGX})}{9 \cdot 10^{-\text{NSIG}} \cdot \cosh x}.$$

Here, μ is $\frac{1}{I_{\text{MAGX}}}(x)$, so the relative error $|R^{(N)}/\mu|$ is bounded by

$$\left| \frac{R^{(N)}}{\mu} \right| \leq \frac{4 I_{\text{MAGX}}(x) \exp(0.461 \cdot \text{MAGX})}{9 \cosh x \cdot 10^{-\text{NSIG}}}. \quad (8)$$

Now Kapteyn's inequality [4, 8.7] states that

$$|J_n(nz)| \leq \left| \frac{z \exp \sqrt{1-z^2}}{1 + \sqrt{1-z^2}} \right|^n.$$

Let $z = \frac{|x|}{\text{MAGX}}$, so for large x , $z = 1$, approximately.

Then

$$\begin{aligned} |I_{\text{MAGX}}(x)| &= |J_{\text{MAGX}}(ix)| = |J_{\text{MAGX}}(\text{MAGX} \cdot iz)| \\ &\leq \left| \frac{z \exp \sqrt{1+z^2}}{1 + \sqrt{1+z^2}} \right|^{\text{MAGX}}. \end{aligned}$$

For large x , the approximate bound for $I_{\text{MAGX}}(x)$ is

$$\left(\frac{\exp \sqrt{2}}{1 + \sqrt{2}} \right)^{\text{MAGX}} < \exp(0.533 \cdot \text{MAGX}).$$

Substituting this in (8), we obtain

$$\begin{aligned} \left| \frac{R^{(N)}}{\mu} \right| &< \frac{4 \cdot \exp(0.533 \text{ MAGX}) \cdot \exp(0.461 \cdot \text{MAGX})}{9 \cosh x \cdot 10^{-\text{NSIG}}} \\ &< \frac{4}{9} \cdot 10^{-\text{NSIG}} < \frac{1}{2} \cdot 10^{-\text{NSIG}}. \end{aligned}$$

Besides this approximate upper bound, the strict bounds

$$\left| \frac{R^{(N)}}{\mu} \right| < 7.6 \cdot 10^{-\text{NSIG}} \quad |x| \geq 1$$

and

$$\left| \frac{R^{(N)}}{\mu} \right| < 0.565 \cdot 10^{-\text{NSIG}} \quad |x| < 1$$

may be derived by using Kapteyn's inequality to obtain a strict bound for $|I_{\text{MAGX}+1}(x)|$.

4. Appendix: Algorithm BESLRI

W ELT BESLRI,1,730222, 58730

```

000001          SUBROUTINE BESLRI(X,NB,IZE,B,NCALC)          00000100
000002          C THIS ROUTINE CALCULATES BESSEL FUNCTIONS I AND J OF REAL 00000200
000003          C ARGUMENT AND INTEGER ORDER.                  00000300
000004          C                                           00000400
000005          C                                           00000500
000006          C          EXPLANATION OF VARIABLES IN THE CALLING SEQUENCE 00000600
000007          C                                           00000700
000008          C X      DOUBLE PRECISION REAL ARGUMENT FOR WHICH I*S OR J*S 00000800
000009          C      ARE TO BE CALCULATED. IF I*S ARE TO BE CALCULATED, 00000900
000010          C      ABS(X) MUST BE LESS THAN EXPARG (WHICH SEE BELOW). 00001000
000011          C NB     INTEGER TYPE. I + HIGHEST ORDER TO BE CALCULATED. 00001100
000012          C      IT MUST BE POSITIVE.                    00001200
000013          C IZE    INTEGER TYPE. ZERO IF J*S ARE TO BE CALCULATED, I 00001300
000014          C      IF I*S ARE TO BE CALCULATED.            00001400
000015          C B      DOUBLE PRECISION VECTOR OF LENGTH NB, NEED NOT BE 00001500
000016          C      INITIALIZED BY USER. IF THE ROUTINE TERMINATES 00001600
000017          C      NORMALLY (NCALC=NB), IT RETURNS J(OOR I)-SUB-ZERO 00001700
000018          C      THROUGH J(OOR I)-SUB-NB-MINUS-ONE OF X IN THIS 00001800
000019          C      VECTOR.                                  00001900
000020          C NCALC   INTEGER TYPE, NEED NOT BE INITIALIZED BY USER. 00002000
000021          C      BEFORE USING THE RESULTS, THE USER SHOULD CHECK THAT 00002100
000022          C      NCALC=NB, I.E. ALL ORDERS HAVE BEEN CALCULATED TO 00002200
000023          C      THE DESIRED ACCURACY. SEE ERROR RETURNS BELOW. 00002300
000024          C                                           00002400
000025          C                                           00002500
000026          C          EXPLANATION OF MACHINE-DEPENDENT CONSTANTS 00002600
000027          C                                           00002700
000028          C NSIG   DECIMAL SIGNIFICANCE DESIRED. SHOULD BE SET TO 00002800
000029          C      IFIX(ALOG10(2)*NBIT+1), WHERE NBIT IS THE NUMBER OF 00002900
000030          C      BITS IN THE MANTISSA OF A DOUBLE PRECISION VARIABLE. 00003000
000031          C      SETTING NSIG LOWER WILL RESULT IN DECREASED ACCURACY 00003100
000032          C      WHILE SETTING NSIG HIGHER WILL INCREASE CPU TIME 00003200
000033          C      WITHOUT INCREASING ACCURACY. THE TRUNCATION ERROR 00003300
000034          C      IS LIMITED TO  $2.5 \cdot 10^{**NSIG}$  FOR J*S OF ORDER LESS 00003400
000035          C      THAN ARGUMENT, AND TO A RELATIVE ERROR OF T FOR 00003500
000036          C      I*S AND THE OTHER J*S.                   00003600
000037          C NTEN   LARGEST INTEGER K SUCH THAT  $10^{**K}$  IS MACHINE- 00003700
000038          C      REPRESENTABLE IN DOUBLE PRECISION.        00003800
000039          C LARGE   UPPER LIMIT ON THE MAGNITUDE OF X. BEAR IN MIND 00003900
000040          C      THAT IF  $ABS(X)=N$ , THEN AT LEAST N ITERATIONS OF THE 00004000
000041          C      BACKWARD RECURSION WILL BE EXECUTED.     00004100
000042          C EXPARG  LARGEST DOUBLE PRECISION ARGUMENT THAT THE LIBRARY 00004200
000043          C      DEXP ROUTINE CAN HANDLE.                 00004300
000044          C                                           00004400
000045          C                                           00004500
000046          C          ERROR RETURNS                          00004600
000047          C                                           00004700
000048          C      LET G DENOTE EITHER I OR J.              00004800
000049          C      IN CASE OF AN ERROR, NCALC.NE.NB, AND NOT ALL G*S 00004900
000050          C      ARE CALCULATED TO THE DESIRED ACCURACY.    00005000
000051          C      IF NCALC.LT.0, AN ARGUMENT IS OUT OF RANGE. NB.LE.0 00005100
000052          C      OR IZE IS NEITHER 0 NOR 1 OR IZE=1 AND  $ABS(X) \cdot GE \cdot EXPARG$ . 00005200
000053          C      IN THIS CASE, THE B-VECTOR IS NOT CALCULATED, AND NCALC 00005300
000054          C      IS SET TO  $MIN0(NB,0)-1$  SO NCALC.NE.NB. 00005400

```

```

000055 C NB,GT,NCALC,GT,0 WILL OCCUR IF NB,GT,MAGX AND ABS(G-
000056 C SUB-NB-OF-X/G-SUB-MAGX-OF-X).LT.10.**(NTEN/2); I. E. NB
000057 C IS MUCH GREATER THAN MAGX. IN THIS CASE, B(N) IS CALCU-
000058 C LATED TO THE DESIRED ACCURACY FOR N,LE,NCALC, BUT FOR
000059 C NCALC.LT.N,LE,NP, PRECISION IS LOST. IF N,GT,NCALC AND
000060 C ABS(B(NCALC)/B(N)).EQ.10**K, THEN ONLY THE FIRST NSIG=K
000061 C SIGNIFICANT FIGURES OF B(N) MAY BE TRUSTED. IF THE USER
000062 C WISHES TO CALCULATE B(N) TO HIGHER ACCURACY, HE SHOULD USE
000063 C AN ASYMPTOTIC FORMULA FOR LARGE ORDER.
000064 C
000065 C DOUBLE PRECISION
000066 C 1 X**P,TEST,TEMPA,TEMPB,TEMPC,EXPARG,SIGN,SUM,TOVER,
000067 C 2 PLAST,POLD,PSAVE,PSAVEL
000068 C DIMENSION B(NB)
000069 C DATA NSIG,NTEN,LARGEX,EXPARG/19,307,10000,7,D2/
000070 C TEMPA=DABS(X)
000071 C MAGX=IFY(SNGL(TEMPA))
000072 C IF(NB,GT,0.AND,MAGX,LE,LARGEX.AND,(IZE,EQ,0.OR,
000073 C 1(IZE,EQ,1.AND,TEMPA,LE,EXPARG))) GO TO 1
000074 C ERROR RETURN -- X,NB,OR IZE IS OUT OF RANGE
000075 C NCALC=MIN(NB,0)-1
000076 C RETURN
000077 C 1 SIGN=DBLE(FLOAT(1-2*IZE))
000078 C NCALC=NB
000079 C USE 2-TERM ASCENDING SERIES FOR SMALL X
000080 C IF(TEMPA**4.LT.,100**NSIG) GO TO 30
000081 C INITIALIZE THE CALCULATION OF P*S
000082 C NRMX=NB-MAGX
000083 C N=MAGX+1
000084 C PLAST=1.D0
000085 C P=DBLE(FLOAT(2*N))/TEMPA
000086 C CALCULATE GENERAL SIGNIFICANCE TEST
000087 C TEST=2.D0*1.D1**NSIG
000088 C IF(IZE,EQ,1.AND,2*MAGX,GT,5*NSIG) TEST=DSQRT(TEST*P)
000089 C IF(IZE,EQ,1.AND,2*MAGX,LE,5*NSIG) TEST=TEST/1.585**MAGX
000090 C #E0
000091 C IF(NRMX.LT,3) GO TO 4
000092 C CALCULATE P*S UNTIL N=NB-1. CHECK FOR POSSIBLE OVERFLOW.
000093 C TOVER=1.D1**(NTEN-NSIG)
000094 C NSTART=MAGX+2
000095 C NEND=NB-1
000096 C DO 3 N=NSTART,NEND
000097 C POLD=PLAST
000098 C PLAST=P
000099 C P=DBLE(FLOAT(2*N))*PLAST/TEMPA-SIGN*POLD
000100 C IF(P>TOVER) 3,3,5
000101 C 3 CONTINUE
000102 C N=NEND
000103 C CALCULATE SPECIAL SIGNIFICANCE TEST FOR NRMX,GT,2.
000104 C TEST=DMAX1(TEST,DSQRT(PLAST*1.D1**NSIG)*DSQRT(2.D0*P))
000105 C CALCULATE P*S UNTIL SIGNIFICANCE TEST PASSES
000106 C 4 N=N+1
000107 C POLD=PLAST
000108 C PLAST=P
000109 C P=DBLE(FLOAT(2*N))*PLAST/TEMPA-SIGN*POLD
000110 C IF(P.LT,TEST) GO TO 4
000111 C IF(IZE,EQ,1.OR,M,EQ,1) GO TO 12
000112 C FOR J*S, A STRONG VARIANT OF THE TEST IS NECESSARY.

```

```

00005500
00005600
00005700
00005800
00005900
00006000
00006100
00006200
00006300
00006400
00006500
00006600
00006700
00006800
00006900
00007000
00007100
00007200
00007300
00007400
00007500
00007600
00007700
00007800
00007900
00008000
00008100
00008200
00008300
00008400
00008500
00008600
00008700
00008800
00008900
00009000
00009100
00009200
00009300
00009400
00009500
00009600
00009700
00009800
00009900
00010000
00010100
00010200
00010300
00010400
00010500
00010600
00010700
00010800
00010900
00011000
00011100
00011200

```

000113	C CALCULATE IT, AND CALCULATE P*S UNTIL THIS TEST IS PASSED.	00011300
000114	M=1	00011400
000115	TEMPB=P/PLAST	00011500
000116	TEMPC=DBLE(FLOAT(N+1))/TEMPA	00011600
000117	IF(TEMPB+1.00/TEMPB.GT.2.00*TEMPC)TEMPR=TEMPC+DSQRT	00011700
000118	1(TEMPC**2-1.00)	00011800
000119	TEST=TEST/DSQRT(TEMPB-1.00/TEMPB)	00011900
000120	IF(P-TEST) 4,12,12	00012000
000121	C TO AVOID OVERFLOW, DIVIDE P*S BY TOVER. CALCULATE P*S	00012100
000122	C UNTIL ABS(P).GT.1.	00012200
000123	5 TOVER=1.01**NTEN	00012300
000124	P=P/TOVER	00012400
000125	PLAST=PLAST/TOVER	00012500
000126	PSAVE=P	00012600
000127	PSAVEL=PLAST	00012700
000128	NSTART=N+1	00012800
000129	6 N=N+1	00012900
000130	POLD=PLAST	00013000
000131	PLASTE=P	00013100
000132	P=DBLE(FLOAT(2*N))*PLAST/TEMPA-SIGN*POLD	00013200
000133	IF(P.LE.1.00) GO TO 6	00013300
000134	TEMPB=DBLE(FLOAT(2*N))/TEMPA	00013400
000135	IF(12E,0,1) GO TO 8	00013500
000136	TEMPC=.500*TEMPB	00013600
000137	TEMPR=PLAST/POLD	00013700
000138	IF(TEMPR+1.00/TEMPB.GT.2.00*TEMPC)TEMPB=TEMPC+DSQRT	00013800
000139	1(TEMPC**2-1.00)	00013900
000140	C CALCULATE BACKWARD TEST, AND FIND NCALC, THE HIGHEST N	00014000
000141	C SUCH THAT THE TEST IS PASSED.	00014100
000142	8 TEST=.500*POLD*PLAST*(1.00-1.00/TEMPB**2)/1.01**NSIG	00014200
000143	P=PLAST*TOVER	00014300
000144	N=N-1	00014400
000145	NEND=MIND(NB,N)	00014500
000146	DO 9 NCALC=NSTART,NEND	00014600
000147	POLD=PSAVEL	00014700
000148	PSAVE=PSAVE	00014800
000149	PSAVE=DBLE(FLOAT(2*N))*PSAVEL/TEMPA-SIGN*POLD	00014900
000150	IF(PSAVE*PSAVEL-TEST) 9,9,10	00015000
000151	9 CONTINUE	00015100
000152	NCALC=NEND+1	00015200
000153	10 NCALC=NCALC-1	00015300
000154	C THE SUM B(1)+2B(3)+2B(5)... IS USED TO NORMALIZE. M, THE	00015400
000155	C COEFFICIENT OF B(N), IS INITIALIZED TO 2 OR 0.	00015500
000156	12 N=N+1	00015600
000157	M=2*N-4*(N/2)	00015700
000158	C INITIALIZE THE BACKWARD RECURSION AND THE NORMALIZATION	00015800
000159	C SUM	00015900
000160	TEMPB=0.00	00016000
000161	TEMPA=1.00/P	00016100
000162	SUM=DBLE(FLOAT(M))*TEMPA	00016200
000163	NEND=N-NB	00016300
000164	IF(NEND) 17,15,13	00016400
000165	C RECUR BACKWARD VIA DIFFERENCE EQUATION, CALCULATING (BUT	00016500
000166	C NOT STORING) B(N), UNTIL N=NB.	00016600
000167	13 DO 14 L=1,NEND	00016700
000168	N=N-1	00016800
000169	TEMPC=TEMPB	00016900
000170	TEMPB=TEMPA	00017000

```

000171          TEMPA=(DBLE(FLOAT(2*N))*TEMPB)/X-SIGN*TEMPC
000172          M=2-M
000173          14  SUM=SUM+DBLE(FLOAT(M))*TEMPA
000174          C STORE B(NB)
000175          15  B(N)=TEMPA
000176          IF(NB.GT.1) GO TO 16
000177          C NB=1. SINCE 2*TEMPA WAS ADDED TO THE SUM, TEMPA MUST BE
000178          C SUBTRACTED
000179          SUM=SUM-TEMPA
000180          GO TO 23
000181          C CALCULATE AND STORE B(NB-1)
000182          16  N=N-1
000183          B(N)=(DBLE(FLOAT(2*N))*TEMPA)/X-SIGN*TEMPB
000184          IF(N.EQ.1) GO TO 22
000185          M=2-M
000186          SUM=SUM+DBLE(FLOAT(M))*B(N)
000187          GO TO 19
000188          C N.LT.NB, SO STORE B(N) AND SET HIGHER ORDERS TO ZERO
000189          17  B(N)=TEMPA
000190          NEND=-NEND
000191          DO 18 L=1,NEND
000192          18  B(L)=0.00
000193          19  NEND=N-2
000194          IF(NEND.EQ.0) GO TO 21
000195          C CALCULATE VIA DIFFERENCE EQUATION AND STORE B(N),
000196          C UNTIL N=2
000197          DO 20 L=1,NEND
000198          N=N-1
000199          B(N)=(DBLE(FLOAT(2*N))*B(N+1))/X-SIGN*B(N+2)
000200          M=2-M
000201          20  SUM=SUM+DBLE(FLOAT(M))*B(N)
000202          C CALCULATE B(1)
000203          21  B(1)=2.00*B(2)/X-SIGN*B(3)
000204          22  SUM=SUM+B(1)
000205          C NORMALIZE--IF IZE=1, DIVIDE SUM BY COSH(X). DIVIDE ALL
000206          C B(N) BY SUM.
000207          23  IF(IZE.EQ.0) GO TO 25
000208          TEMPA=DEXP(DABS(X))
000209          SUM=2.00*SUM/(TEMPA+1.00/TEMPA)
000210          25  DO 26 N=1,N3
000211          26  B(N)=B(N)/SUM
000212          RETURN
000213          C TWO-TERM ASCENDING SERIES FOR SMALL X
000214          30  TEMPA=1.00
000215          TEMPB=-.2500*X*X*SIGN
000216          B(1)=1.00+TEMPB
000217          IF(N3.EQ.1) GO TO 32
000218          DO 31 N=2,N3
000219          TEMPA=TEMPA*X/DBLE(FLOAT(2*N-2))
000220          31  B(N)=TEMPA*(1.00+TEMPB/DBLE(FLOAT(N)))
000221          32  RETURN
000222          END
3.

```

END CUR LCC 1102-0038 LB

5. References

- [1] Olver, F. W. J., and Sookne, D. J., A note on backward recurrence algorithms, *Mathematics of Computation*, Vol. 26, No. 120 (Oct. 1972), pp. 941-947.
- [2] Olver, F. W. J., Numerical solution of second-order linear difference equations, *J. Res. Nat. Bur. Stand. (U.S.)*, 71B, (Math. and Math. Phys.), Nos 2&3, 111-129 (Apr.-Sept. 1967).
- [3] British Assoc. for the Advancement of Science, *Bessel Functions—Part II*, *Mathematical Tables*, Vol. 10 (Cambridge University Press, Cambridge, 1952).
- [4] Watson, G. N., *A Treatise on the Theory of Bessel Functions* (Second Edition, Cambridge University Press, Cambridge, 1944).

(Paper 77B3&4-387)