

Certification of an Algorithm for Bessel Functions of Real Argument

David J. Sookne*

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

(June 4, 1973)

The accuracy of routine BESLRI is certified.

Key words: Bessel functions; bit comparison; error bounds.

1. Introduction

Bessel function values $I_n(x)$ and $J_n(x)$ (x real), generated by BESLRI [1],¹ were compared with check values. For $|x| < 64$, these check values were calculated via the ascending series

$$J_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \frac{(-x^2/4)^k}{k!(n+k)!}, \quad I_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \frac{(x^2/4)^k}{k!(n+k)!}$$

using multiprecision arithmetic [2]. For $x \geq 64$, the asymptotic formulas for large argument were used:

$$I_n(x) \sim \frac{e^x}{\sqrt{2\pi x}} \left(1 - \frac{(\mu-1)}{8x} + \frac{(\mu-1)(\mu-9)}{2!(8x)^2} - \frac{(\mu-1)(\mu-9)(\mu-25)}{3!(8x)^3} + \dots\right)$$

where $\mu = 4n^2$, and error is controlled by 7.16 of [3], and

$$J_n(x) = \sqrt{\frac{2}{\pi x}} \{P(n, x) \cos \chi - Q(n, x) \sin \chi\}$$

where $\chi = x - \left(\frac{n}{2} + \frac{1}{4}\right)\pi$, and

$$P(n, x) \sim 1 - \frac{(\mu-1)(\mu-9)}{2!(8x)^2} + \frac{(\mu-1)(\mu-9)(\mu-25)(\mu-49)}{4!(8x)^4} + \dots$$

$$Q(n, x) \sim \frac{\mu-1}{8x} - \frac{(\mu-1)(\mu-9)(\mu-25)}{3!(8x)^3} + \dots$$

AMS Subject Classification: 68A10.

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

¹ Figures in brackets indicate the literature references at the end of this paper.

with errors bounded by the first neglected term [4, 9.2.9 and 9.2.10]. These were programmed in double precision. For $x \leq -64$, $-x$ was used as the argument, and the sign of the function value was determined from the equations

$$J_n(-x) = (-1)^n J_n(x), \quad I_n(-x) = (-1)^n I_n(x).$$

The method used was that of bit comparison; the difference between the 60-bit mantissas of test and check values was expressed as a multiple, m , say, of the last bit. Such a bit error corresponds to a relative error between $m \cdot 2^{-60}$ and $m \cdot 2^{-59}$.

This test is too strict near a zero of $J_n(x)$, where absolute error is a more realistic measure. Thus for $n < |x|$, both mantissas were right-shifted so that the bit error described above was a multiple of the 60th binary place. These cases are given above the dashed line in the computer printout; statistics for this test are compiled separately.

A bit comparison test was also used to check the section of the code involving the two-term ascending series for small $|x|$.

The error return feature was tested exactly as in [5].

2. The Bit Comparison Test

To test the function $I_n(x)$, ten arguments were chosen in each interval

$$A_j = \{x: 2^{j-1} \leq |x| < 2^j\}, \quad j = -13(1)10,$$

the 60-bit mantissas of each x being produced by a pseudorandom number generator [5]. The orders used in all cases were $n = 0(1)15$. The only modification occurred in the set A_{10} , for which $|x|$ was limited to 700 to avoid computer overflow in the final answers. Summary statistics for the 60-bit test on I_n are:

373	results correct to 60 bits
583	results in error by 1 multiple of the last bit
383	results in error by 2 multiples of the last bit
288	results in error by 3 multiples of the last bit
232	results in error by 4 multiples of the last bit
196	results in error by 5 multiples of the last bit
176	results in error by 6 multiples of the last bit
130	results in error by 7 multiples of the last bit
1479	results in error by more than 7 multiples of the last bit
<hr/>	
3840	(total)

The largest bit error was (36)₈, implying that at least 55 bits are correct in each case.

For the function $J_n(x)$, the set of intervals A_j was expanded to $j = -13(1)16$. Again, ten pseudorandom values were used in each interval, and the orders used were $0(1)15$. Summaries for the tests are:

Test on 60 Significant Bits $|x| < 64$

403	results correct to 60 significant bits
643	results in error by 1 multiple of the last bit
453	results in error by 2 multiples of the last bit
331	results in error by 3 multiples of the last bit

232 results in error by 4 multiples of the last bit
 171 results in error by 5 multiples of the last bit
 109 results in error by 6 multiples of the last bit
 68 results in error by 7 multiples of the last bit
 82 results in error by more than 7 multiples of the last bit

2492 (total)

The largest bit error was (15)₈, showing that at least 56 bits are correct in each case.

Test on 60 Binary Places $|x| < 64$

161 results correct to 60 places
 200 results in error by 1 multiple of the last place
 100 results in error by 2 multiples of the last place
 85 results in error by 3 multiples of the last place
 56 results in error by 4 multiples of the last place
 46 results in error by 5 multiples of the last place
 45 results in error by 6 multiples of the last place
 14 results in error by 7 multiples of the last place
 1 result in error by more than 7 multiples of the last place.

708 (total)

The largest bit error was (10)₈, showing that 56 binary places are correct in each case, and 57 places in all but one case.

In the test on 60 binary places for $|x| \geq 64$, the largest error was (422)₈. This occurred in $J_5(x)$ and $J_7(x)$ for $x = 59766.6 \dots$. Bearing in mind that about 60,000 back-recursion steps were executed, that each step included two multiplications and one addition, and also that the computer uses truncation in floating-point arithmetic rather than roundoff, we see that the error of (422)₈ = 274 is quite reasonable. The other errors can be explained similarly.

For the test of the two-term ascending series, arguments of 2^{-16} and 2^{-100} were used; the former is just barely small enough for the truncation error to be neglected. Both $I_n(x)$ and $J_n(x)$ were tested with $n = 0(1)15$, and the largest error was 6 multiples of the last bit.

3. Testing of Error Return

The method used is the same as in [5]. Arguments used were $x = 2^k$ where $k = -13(1)9$ for I 's, and $k = -13(1)13$ for J 's. In the test on Bessel function values of order NCALC-1, the largest error was 6 multiples of the last bit; most were zero.

In the test on order $[x]$, the largest error was (46)₈, which occurred in $J_{8192}(8192)$. This test value involved about 1600 more back-recursion steps than were required for the check value, so this error is quite reasonable. The other errors were (35)₈ for $x = 4096$, zero for $x = 2048$, (12)₈ for $x = 1024$; for $x \leq 512$, the errors did not exceed 5.

4. Summary and Conclusions

For $n = 0(1)15$ and the values of x tested, BESLRI yielded $J_n(x)$ and $I_n(x)$ with either a relative error or an absolute error bounded by 10^{-14} . The latter assessment applied to $J_n(x)$ when $n < |x|$. For $|x| < 64$, the bound was 10^{-16} . The error return and ascending series features were tested and found to be executed accurately in all cases.

For other values of n and x , similar accuracy can be expected, since the tests we have used include large and small values of $|x|$ and $\frac{|x|}{n}$, the former as large as 2^{16} .

Calculations were performed on a UNIVAC 1108 under EXEC 2. All test values were calculated with double precision mantissa of 60 bits, corresponding to just over 18 figures. Subroutine executions took approximately $0.2 N$ milliseconds, where

$$N = \max(|x|, NB) + 10.$$

5. Appendix: Algorithm BESLRI

© ELT BESLRI, 1,730112, 51748

```

000001          SUBROUTINE BESLRI(X,Y,NB,IZE,BR,BI,NCALC)          00010
000002          C THIS ROUTINE CALCULATES BESSEL FUNCTIONS I AND J OF 00020
000003          C COMPLEX ARGUMENT AND INTEGER ORDER.                00030
000004          C                                                       00040
000005          C                                                       00050
000006          C          EXPLANATION OF VARIABLES IN THE CALLING SEQUENCE 00060
000007          C                                                       00070
000008          C X          DOUBLE PRECISION REAL PART OF THE COMPLEX ARGUMENT 00080
000009          C          FOR WHICH I*S OR J*S ARE TO BE CALCULATED. IF I*S 00090
000010          C          ARE TO BE CALCULATED, ABS(X) MUST NOT EXCEED EXPARG 00100
000011          C          (WHICH SEE BELOW).                               00110
000012          C Y          IMAGINARY PART OF THE ARGUMENT. IF J*S ARE TO BE 00120
000013          C          CALCULATED, ABS(Y) MUST NOT EXCEED EXPARG.     00130
000014          C NB         INTEGER TYPE. 1 * HIGHEST ORDER TO BE CALCULATED. 00140
000015          C          IT MUST BE POSITIVE.                           00150
000016          C IZE       INTEGER TYPE. ZERO IF J*S ARE TO BE CALCULATED, 1 00160
000017          C          IF I*S ARE TO BE CALCULATED.                   00170
000018          C BR         DOUBLE PRECISION VECTOR OF LENGTH NB, NEED NOT BE 00180
000019          C          INITIALIZED BY USER. IF THE ROUTINE TERMINATES 00190
000020          C          NORMALLY, (NCALC*NB), IT RETURNS THE REAL PART OF 00200
000021          C          J(OR I)-SUB-ZERO THROUGH J(OR I)-SUB-NB-MINUS-ONE 00210
000022          C          OF Z IN THIS VECTOR.                            00220
000023          C BI         IMAGINARY ANALOG OF BR.                       00230
000024          C NCALC      INTEGER TYPE, NEED NOT BE INITIALIZED BY USER. 00240
000025          C          BEFORE USING THE RESULTS, THE USER SHOULD CHECK THAT 00250
000026          C          NCALC*NB, I.E. ALL ORDERS HAVE BEEN CALCULATED TO 00260
000027          C          THE DESIRED ACCURACY. SEE ERROR RETURNS BELOW. 00270
000028          C                                                       00280
000029          C                                                       00290
000030          C          EXPLANATION OF MACHINE-DEPENDENT CONSTANTS      00300
000031          C                                                       00310
000032          C NSIG       DECIMAL SIGNIFICANCE DESIRED. SHOULD BE SET TO 00320
000033          C          IFIX(ALOG10(2)*NBIT*1), WHERE NBIT IS THE NUMBER OF 00330
000034          C          BITS IN THE MANTISSA OF A DOUBLE PRECISION VARIABLE. 00340
000035          C          SETTING NSIG HIGHER WILL INCREASE CPU TIME WITHOUT 00350
000036          C          INCREASING ACCURACY, WHILE SETTING NSIG LOWER WILL 00360
000037          C          DECREASE ACCURACY. IF ONLY SINGLE-PRECISION 00370
000038          C          ACCURACY IS DESIRED, REPLACE NBIT BY THE NUMBER OF 00380
000039          C          BITS IN THE MANTISSA OF A SINGLE-PRECISION VARIABLE. 00390
000040          C          THE RELATIVE TRUNCATION ERROR IS LIMITED TO T*.5*10 00400
000041          C          **-NSIG FOR ORDER GREATER THAN ABS(Z), AND FOR ORDER 00410
000042          C          LESS THAN ABS(Z) (GENERAL TEST), THE RELATIVE ERROR 00420
000043          C          IS LIMITED TO T FOR FUNCTION VALUES OF MAGNITUDE AT 00430
000044          C          LEAST 1, AND THE ABSOLUTE ERROR IS LIMITED TO T FOR 00440
000045          C          SMALLER VALUES.                                 00450
000046          C NNTEN      LARGEST INTEGER K SUCH THAT 10**K IS MACHINE- 00460
000047          C          REPRESENTABLE IN DOUBLE PRECISION.              00470
000048          C LARGEZ     UPPER LIMIT ON THE MAGNITUDE OF Z. BEAR IN MIND 00480
000049          C          THAT IF ABS(Z)*N, THEN AT LEAST N ITERATIONS OF THE 00490
000050          C          BACKWARD RECURSION WILL BE EXECUTED.            00500
000051          C EXPARG     LARGEST DOUBLE PRECISION ARGUMENT THAT THE LIBRARY 00510
000052          C          DEXP ROUTINE CAN HANDLE.                          00520
000053          C                                                       00530
000054          C                                                       00540

```

J. Res. 77B3&4-386

```

000055 C                                     ERROR RETURNS                                00550
000056 C                                     00560
000057 C      LET G DENOTE EITHER I OR J.                                00570
000058 C      IN CASE OF AN ERROR, NCALC,NE,NB, AND NOT ALL G*S          00580
000059 C ARE CALCULATED TO THE DESIRED ACCURACY.                        00590
000060 C      IF NCALC,LT,0, AN ARGUMENT IS OUT OF RANGE. NB,LE,0       00600
000061 C OR IZE IS NEITHER 0 NOR 1 OR IZE=0 AND ABS(Y),GT,EXPARG,       00610
000062 C OR IZE=1 AND ABS(X),GT,EXPARG. IN THIS CASE, THE VECTORS      00620
000063 C BR AND BI ARE NOT CALCULATED, AND NCALC IS SET TO            00630
000064 C MINO(NB,0)-1 SO NCALC,NE,NB,                                00640
000065 C NB,GT,NCALC,GT,0 WILL OCCUR IF NE,GT,MAGZ AND ABS(G-          00650
000066 C SUB-NB-OF-Z/G-SUB-MAGZ-OF-Z),LT,10.**(NTEN/2), I. E. NB      00660
000067 C IS MUCH GREATER THAN MAGZ. IN THIS CASE, BR(N) AND BI(N)     00670
000068 C ARE CALCULATED TO THE DESIRED ACCURACY FOR N,LE,NCALC,       00680
000069 C BUT FOR NCALC,LT,N,LE,NB, PRECISION IS LOST. IF N,GT,       00690
000070 C NCALC AND ABS(G(NCALC-1)/G(N-1)),EQ,10**=-K, THEN THE LAST  00700
000071 C K SIGNIFICANT FIGURES OF G(N-1) (-BR(N)*I*BI(N)) ARE        00710
000072 C ERRONEOUS. IF THE USER WISHES TO CALCULATE G(N-1) TO       00720
000073 C HIGHER ACCURACY, HE SHOULD USE AN ASYMPTOTIC FORMULA FOR   00730
000074 C LARGE ORDER.                                               00740
000075 C                                                                00750
000076 C      DOUBLE PRECISION                                         00760
000077 C      1 X,Y,BR,BI,PR,PI,PLASTR,PLASTI,POLDR,POLDI,PSAVER,      00770
000078 C      2 PSAVEL,EXPARG,TEST,TOVER,TEMPAR,TEMPAI,TEMPBR,TEMPBI,  00780
000079 C      3 TEMPCR,TEMPCI,SIGN,SUMR,SUMI,ZINVR,ZINVI              00790
000080 C      DIMENSION BR(NB),BI(NB)                                00800
000081 C      DATA NSIG,NTEN,LARGEZ,EXPARG/19,307,10000,7.D2/        00810
000082 C      TEMPAR=DSORT(X*X+Y*Y)                                  00820
000083 C      MAGZ=FIX(SNGL(TEMPAR))                                  00830
000084 C      IF(NB,GT,0.AND.MAGZ,LE,LARGEZ.AND.((IZE,EQ,0.AND.       00840
000085 C      1 DABS(Y),LE,EXPARG).OR.(IZE,EQ,1.AND,DAES(X),LE.      00850
000086 C      2 EXPARG))) GO TO 1                                     00860
000087 C ERROR RETURN -- Z, NB, OR IZE IS OUT OF RANGE              00870
000088 C      NCALC=MINO(NB,0)-1                                     00880
000089 C      RETURN                                                00890
000090 C      1 SIGN=DBLE(FL0AT(1-2*IZE))                             00900
000091 C      NCALC=NB                                              00910
000092 C USE 2-TERM ASCENDING SERIES FOR SMALL Z                    00920
000093 C      IF(TEMPAR**4,LT,.1D0**NSIG) GO TO 50                 00930
000094 C INITIALIZE THE CALCULATION OF THE P*S                      00940
000095 C      NBMZ=NB-MAGZ                                          00950
000096 C      N=MAGZ+1                                             00960
000097 C      IF(DABS(X),LT,DABS(Y)) GO TO 2                       00970
000098 C      ZINVR=1.D0/(X+Y*X/Y)                                  00980
000099 C      ZINVI=-Y*ZINVR/X                                       00990
000100 C      GO TO 3                                                01000
000101 C      2 ZINVI=-1.D0/(Y*X*X/Y)                                01010
000102 C      ZINVR=-X*ZINVI/Y                                       01020
000103 C      3 PLASTR=1.D0                                         01030
000104 C      PLASTI=0.D0                                           01040
000105 C      PR=SIGN*DBLE(FL0AT(2*N))*ZINVR                          01050
000106 C      PI=SIGN*DBLE(FL0AT(2*N))*ZINVI                          01060
000107 C      TEST=2.D0*1.D1**NSIG                                  01070
000108 C      M=0                                                    01080
000109 C      IF(NBMZ,LT,3) GO TO 6                                   01090
000110 C CALCULATE P*S UNTIL N=NB-1. CHECK FOR POSSIBLE OVERFLOW.  01100
000111 C      TOVER=1.D1***(NTEN-NSIG)                              01110
000112 C      NSTART=MAGZ+2                                         01120

```

000113	NEND=NB-1	01130
000114	D0 5 N*NBSTART,NEND	01140
000115	P0LDR=PLASTR	01150
000116	P0LDI=PLASTI	01160
000117	PLASTR=PR	01170
000118	PLASTI=PI	01180
000119	PR=SIGN*(DBLE(FL0AT(2*N)))*(PLASTR*ZINVR-PLASTI*ZINVI	01190
000120)-P0LDR)	01200
000121	1 PI=SIGN*(DBLE(FL0AT(2*N)))*(PLASTI*ZINVR+PLASTR*ZINVI	01210
000122)-P0LDI)	01220
000123	1 IF((PR/T0VER)**2*(PI/T0VER)**2-1.D0) 5,5,7	01230
000124	5 CONTINUE	01240
000125	N=NEND	01250
000126	C CALCULATE SPECIAL SIGNIFICANCE TEST FOR NBMZ.GT.2.	01260
000127	TEMPBI=DMAX1(DABS(PR),DABS(PI))	01270
000128	TEMPBI=TEMPBI*DSQRT(2.D0*1.D1**NSIG*DSQRT(((PR/TEMPBI)	01280
000129	1**2*(PI/TEMPBI)**2)*((PLASTR/TEMPBI)**2*(PLASTI/	01290
000130	2TEMPBI)**2))	01300
000131	TEST=DMAX1(TEST,TEMPBI)	01310
000132	C CALCULATE P*S UNTIL SIGNIFICANCE TEST IS PASSED.	01320
000133	6 N=N*1	01330
000134	P0LDR=PLASTR	01340
000135	P0LDI=PLASTI	01350
000136	PLASTR=PR	01360
000137	PLASTI=PI	01370
000138	PR=SIGN*(DBLE(FL0AT(2*N)))*(PLASTR*ZINVR-PLASTI*ZINVI	01380
000139	1 -P0LDR)	01390
000140	PI=SIGN*(DBLE(FL0AT(2*N)))*(PLASTI*ZINVR+PLASTR*ZINVI	01400
000141	1 -P0LDI)	01410
000142	IF((PR/TEST)**2*(PI/TEST)**2.LT.1.D0) G0 T0 6	01420
000143	IF(M.EQ.1) G0 T0 12	01430
000144	C CALCULATE STRICT VARIANT OF SIGNIFICANCE TEST, AND	01440
000145	C CALCULATE P*S UNTIL THIS TEST IS PASSED.	01450
000146	M=1	01460
000147	TEMPBI=DMAX1(DABS(PR),DABS(PI))	01470
000148	TEMPBR=DSQRT(((PR/TEMPBI)**2*(PI/TEMPBI)**2)/	01480
000149	1 (((PLASTR/TEMPBI)**2*(PLASTI/TEMPBI)**2))	01490
000150	TEMPBI=DBLE(FL0AT(N*1))/TEMPAR	01500
000151	IF(TEMPBR+1.D0/TEMPBR.GT.2.D0*TEMPBI) TEMPBR=TEMPBI	01510
000152	1 *DSQRT(TEMPBI**2-1.D0)	01520
000153	TEST=TEST/DSQRT(TEMPBR-1.D0/TEMPBR)	01530
000154	IF((PR/TEST)**2*(PI/TEST)**2-1.D0) 6,12,12	01540
000155	7 NBSTART=N*1	01550
000156	C T0 AVOID OVERFLOW, NORMALIZE P*S BY DIVIDING BY T0VER.	01560
000157	C CALCULATE P*S UNTIL UNNORMALIZED P WOULD OVERFLOW.	01570
000158	PR=PR/T0VER	01580
000159	PI=PI/T0VER	01590
000160	PLASTR=PLASTR/T0VER	01600
000161	PLASTI=PLASTI/T0VER	01610
000162	PSAVER=PR	01620
000163	PSAVER=PI	01630
000164	TEMPCR=PLASTR	01640
000165	TEMPCI=PLASTI	01650
000166	TEST=1.D1**((2*NSIG)	01660
000167	8 N=N*1	01670
000168	P0LDR=PLASTR	01680
000169	P0LDI=PLASTI	01690
000170	PLASTR=PR	01700

```

000171          PLASTI=PI                                01710
000172          PR=SIGN*(DBLE(FLGAT(2*N)))*(PLASTR*ZINVR-PLASTI*ZINVI) 01720
000173          1 -POLDR)                                01730
000174          PI=SIGN*(DBLE(FLGAT(2*N)))*(PLASTI*ZINVR+PLASTR*ZINVI) 01740
000175          1 -POLDI)                                01750
000176          IF(PR**2*PI**2.LE.TEST) GO TO 8          01760
000177          C CALCULATE BACKWARD TEST, AND FIND NCALC, THE HIGHEST N 01770
000178          C SUCH THAT THE TEST IS PASSED.          01780
000179          TEMPBR=DSQRT((PLASTR**2*PLASTI**2)/(POLDR**2*POLDI**2) 01790
000180          1)                                         01800
000181          TEMPBI=DBLE(FLGAT(N))/TEMPAR             01810
000182          IF(TEMPBR*1.DO/TEMPBR.GT.2.DO*TEMPBI) TEMPBR=TEMPBI* 01820
000183          1 DSQRT(TEMPBI**2-1.DO)                   01830
000184          TEST=.5DO*(1.DO-1.DO/TEMPBR**2)/1.D1**NSIG 01840
000185          TESTI=((PLASTR**2*PLASTI**2)*TEST)*((POLDR**2*POLDI**2) 01850
000186          1 *TEST)                                   01860
000187          PR=PLASTR*TOVER                            01870
000188          PI=PLASTI*TOVER                            01880
000189          N=N-1                                      01890
000190          NEND=MINO(NB,N)                            01900
000191          DO 9 NCALC=NSTART,NEND                    01910
000192          POLDR=TEMPCR                               01920
000193          POLDI=TEMPCI                               01930
000194          TEMPCR=PSAVER                             01940
000195          TEMPPI=PSAVEI                              01950
000196          PSAVER=SIGN*(DBLE(FLGAT(2*N)))*(TEMPCR*ZINVR-TEMPCI* 01960
000197          1 ZINVI)-POLDR)                           01970
000198          PSAVEI=SIGN*(DBLE(FLGAT(2*N)))*(TEMPCI*ZINVR+TEMPCR* 01980
000199          1 ZINVI)-POLDI)                           01990
000200          IF((PSAVER**2*PSAVEI**2)*(TEMPCR**2*TEMPCI**2)-TEST) 02000
000201          1 9,9,10                                  02010
000202          9 CONTINUE                                02020
000203          NCALC=NEND+1                              02030
000204          10 NCALC=NCALC-1                          02040
000205          C THE COEFFICIENT OF B(N) IN THE NORMALIZATION SUM IS 02050
000206          C M=SQRT(-1)**IMAG, WHERE M=-2,0, OR 2, AND IMAG IS 0 OR 1. 02060
000207          C CALCULATE RECURSION RULES FOR M AND IMAG, AND INITIALIZE 02070
000208          C THEM.                                    02080
000209          12 N=N+1                                   02090
000210          TEMPBR=DBLE(FLGAT(IZE))*X*DBLE(FLGAT(1-IZE))*Y 02100
000211          IPGS=0                                     02110
000212          IF(TEMPBR) 13,14,13                       02120
000213          13 IPGS=FIX(SNGL(1.1DO*TEMPBR/DABS(TEMPBR))) 02130
000214          14 MRECUR=4*((L*IZE+IPGS)/2)-3-2*(IZE+IPGS) 02140
000215          K=2*IPGS*2*IZE-IPGS**2-IZE              02150
000216          L=N-4*(N/4)                               02160
000217          MLAST=2*8*((K*L)/4)-4*((K*L)/2)          02170
000218          IF(IPGS.EQ.0.AND.(L.EQ.1.OR.L.EQ.3)) MLAST=0 02180
000219          L=L+3-4*((L+3)/4)                          02190
000220          M =2*8*((K*L)/4)-4*((K*L)/2)             02200
000221          IF(IPGS.EQ.0.AND.(L.EQ.1.OR.L.EQ.3)) M=0 02210
000222          IMRECR=(1-IZE)*IPGS**2                   02220
000223          IMAG=IMRECR*(L-2*(L/2))                   02230
000224          C INITIALIZE THE BACKWARD RECURSION AND THE NORMALIZATION 02240
000225          C SUM.                                     02250
000226          TEMPBR=0.DO                                02260
000227          TEMPBI=0.DO                                02270
000228          IF(DABS(PI).GT.DABS(PR)) GO TO 15        02280

```

000229	TEMPAR=1.DO/(PR*PI*(PI/PR))	02290
000230	TEMPAI=-(PI*TEMPAR)/PR	02300
000231	G0 T0 16	02310
000232	15 TEMPAR=-1.DO/(PI*PR*(PR/PI))	02320
000233	TEMPAR=-(PR*TEMPAI)/PI	02330
000234	16 IF (IMAG.NE.0) G0 T0 17	02340
000235	SUMR=DBLE(FL0AT(M))*TEMPAR	02350
000236	SUMI=DBLE(FL0AT(M))*TEMPAI	02360
000237	G0 T0 18	02370
000238	17 SUMR=-DBLE(FL0AT(M))*TEMPAI	02380
000239	SUMI=DBLE(FL0AT(M))*TEMPAR	02390
000240	18 NEND=N-NB	02400
000241	IF(NEND) 26,22,19	02410
000242	C RECUR BACKWARD VIA DIFFERENCE EQUATION CALCULATING (BUT	02420
000243	C NOT STORING) BR(N) AND BI(N) UNTIL N=NB.	02430
000244	19 D0 21 L=1,NEND	02440
000245	N=N-1	02450
000246	TEMPCR=TEMPBR	02460
000247	TEMPCI=TEMPBI	02470
000248	TEMPBR=TEMPAR	02480
000249	TEMPBI=TEMPAI	02490
000250	PR=DBLE(FL0AT(2*N))*ZINVR	02500
000251	PI=DBLE(FL0AT(2*N))*ZINVI	02510
000252	TEMPAR=PR*TEMPBR-PI*TEMPBI-SIGN*TEMPCR	02520
000253	TEMPAI=PR*TEMPBI+PI*TEMPBR-SIGN*TEMPCI	02530
000254	IMAG=(1-IMAG)*IMRECR	02540
000255	K=MLAST	02550
000256	MLAST=M	02560
000257	M*K=RECUR	02570
000258	IF (IMAG.NE.0) G0 T0 20	02580
000259	SUMR=SUMR+DBLE(FL0AT(M))*TEMPAR	02590
000260	SUMI=SUMI+DBLE(FL0AT(M))*TEMPAI	02600
000261	G0 T0 21	02610
000262	20 SUMR=SUMR-DBLE(FL0AT(M))*TEMPAI	02620
000263	SUMI=SUMI+DBLE(FL0AT(M))*TEMPAR	02630
000264	21 CONTINUE	02640
000265	C STORE BR(NB), BI(NB)	02650
000266	22 BR(N)=TEMPAR	02660
000267	BI(N)=TEMPAI	02670
000268	IF(N.GT.1) G0 T0 23	02680
000269	C NB=1. SINCE 2*TEMPAR AND 2*TEMPAI WERE ADDED T0 SUMR AND	02690
000270	C SUMI RESPECTIVELY, WE MUST SUBTRACT TEMPAR AND TEMPAI	02700
000271	SUMR=SUMR-TEMPAR	02710
000272	SUMI=SUMI-TEMPAI	02720
000273	G0 T0 35	02730
000274	C CALCULATE AND STORE BR(NB-1), BI(NB-1)	02740
000275	23 N=N-1	02750
000276	PR=DBLE(FL0AT(2*N))*ZINVR	02760
000277	PI=DBLE(FL0AT(2*N))*ZINVI	02770
000278	BR(N)=PR*TEMPAR-PI*TEMPAI-SIGN*TEMPBR	02780
000279	BI(N)=PR*TEMPAI+PI*TEMPAR-SIGN*TEMPBI	02790
000280	IF(N.EQ.1) G0 T0 34	02800
000281	IMAG=(1-IMAG)*IMRECR	02810
000282	K=MLAST	02820
000283	MLAST=M	02830
000284	M*K=RECUR	02840
000285	IF (IMAG.NE.0) G0 T0 24	02850
000286	SUMR=SUMR+DBLE(FL0AT(M))*BR(N)	02860

000287	SUMI-SUMI*DBLE(FLOAT(M))*BI(N)	02870
000288	G0 T0 30	02880
000289	24 SUMR-SUMR-DBLE(FLOAT(M))*BI(N)	02890
000290	SUMI-SUMI*DBLE(FLOAT(M))*BR(N)	02900
000291	G0 T0 30	02910
000292	C N.LT,NB, S0 STORE BR(N), BI(N), AND SET HIGHER ORDERS ZERO	02920
000293	26 BR(N)=TEMPAR	02930
000294	BI(N)=TEMPAI	02940
000295	NEND=-NEND	02950
000296	D0 27 L-1,NEND	02960
000297	BR(N*L)*0.D0	02970
000298	27 BI(N*L)*0.D0	02980
000299	30 NEND=N-2	02990
000300	IF(NEND.EQ.0) G0 T0 33	03000
000301	C CALCULATE VIA DIFFERENCE EQUATION AND STORE BR(N),BI(N),	03010
000302	C UNTIL N=2	03020
000303	D0 32 L=1,NEND	03030
000304	N*N-1	03040
000305	PR*DBLE(FLOAT(2*N))*ZINVR	03050
000306	PI*DBLE(FLOAT(2*N))*ZINVI	03060
000307	BR(N)*PR*BR(N*1)-PI*BI(N*1)-SIGN*BR(N*2)	03070
000308	BI(N)*PR*BI(N*1)+PI*BR(N*1)-SIGN*BI(N*2)	03080
000309	IMAG*(1-IMAG)*IMRECR	03090
000310	K*MLAST	03100
000311	MLAST*M	03110
000312	M*K*IMRECUR	03120
000313	IF(IMAG.NE.0) G0 T0 31	03130
000314	SUMR-SUMR*DBLE(FLOAT(M))*BR(N)	03140
000315	SUMI-SUMI*DBLE(FLOAT(M))*BI(N)	03150
000316	G0 T0 32	03160
000317	31 SUMR-SUMR-DBLE(FLOAT(M))*BI(N)	03170
000318	SUMI-SUMI*DBLE(FLOAT(M))*BR(N)	03180
000319	32 CONTINUE	03190
000320	C CALCULATE AND STORE BR(1), BI(1)	03200
000321	33 BR(1)=2.D0*(BR(2)*ZINVR-BI(2)*ZINVI)-SIGN*BR(3)	03210
000322	BI(1)=2.D0*(BR(2)*ZINVI+BI(2)*ZINVR)-SIGN*BI(3)	03220
000323	34 SUMR-SUMR*BR(1)	03230
000324	SUMI-SUMI*BI(1)	03240
000325	C CALCULATE NORMALIZATION FACTOR, TEMPAR *I*TEMPAI	03250
000326	35 IF(IZE.EQ.1) G0 T0 36	03260
000327	TEMPCR*DBLE(FLOAT(IP0S))*Y	03270
000328	TEMPCI*DBLE(FLOAT(-IP0S))*X	03280
000329	G0 T0 37	03290
000330	36 TEMPCR*DBLE(FLOAT(IP0S))*X	03300
000331	TEMPCI*DBLE(FLOAT(IP0S))*Y	03310
000332	37 TEMPCR=DEXP(TEMPCR)	03320
000333	TEMPBR=DCOS(TEMPCI)	03330
000334	TEMPBI=DSIN(TEMPCI)	03340
000335	IF(DABS(SUMR).LT.DABS(SUMI)) G0 T0 38	03350
000336	TEMPCI=SUMI/SUMR	03360
000337	TEMPCR=(TEMPCR/SUMR)/(1.D0+TEMPCI*TEMPCI)	03370
000338	TEMPAR=TEMPCR*(TEMPBR+TEMPBI*TEMPCI)	03380
000339	TEMPAI=TEMPCR*(TEMPBI-TEMPBR*TEMPCI)	03390
000340	G0 T0 39	03400
000341	38 TEMPCI=SUMR/SUMI	03410
000342	TEMPCR=(TEMPCR/SUMI)/(1.D0+TEMPCI*TEMPCI)	03420
000343	TEMPAR=TEMPCR*(TEMPBR+TEMPCI*TEMPBI)	03430
000344	TEMPAI=TEMPCR*(TEMPBI*TEMPCI-TEMPBR)	03440
000345	C NORMALIZE	03450
000346	39 D0 40 N=1,NB	03460
000347	TEMPBR=BR(N)*TEMPAR-BI(N)*TEMPAI	03470
000348	BI(N)*BR(N)*TEMPAI*BI(N)*TEMPAR	03480
000349	40 BR(N)*TEMPBR	03490
000350	RETURN	03500
000351	C TWO-TERM ASCENDING SERIES FOR SMALL Z	03510
000352	50 TEMPAR=1.D0	03520
000353	TEMPAI=0.D0	03530
000354	TEMPCR=.25D0*(X*X-Y*Y)	03540
000355	TEMPCI=.5D0*X*Y	03550
000356	BR(1)=1.D0-SIGN*TEMPCR	03560
000357	BI(1)=-SIGN*TEMPCI	03570
000358	IF(NB.EQ.1) G0 T0 52	03580
000359	D0 51 N=2,NB	03590
000360	TEMPBR=(TEMPAR*X-TEMPAI*Y)/DBLE(FLOAT(2*N-2))	03600
000361	TEMPAI=(TEMPAR*Y+TEMPAI*X)/DBLE(FLOAT(2*N-2))	03610
000362	TEMPAR=TEMPBR	03620
000363	TEMPBR=DBLE(FLOAT(N))	03630
000364	BR(N)=TEMPAR*(1.D0-SIGN*TEMPCR/TEMPBR)	03640
000365	1 *TEMPAI+TEMPCI/TEMPBR	03650
000366	51 BI(N)=TEMPAI*(1.D0-SIGN*TEMPCR/TEMPBR)	03660
000367	1 -TEMPAR*TEMPCI/TEMPBR	03670
000368	52 RETURN	03680
000369	END	03690

END CUR LCC 1102-0038 L8

6. References

- [1] Algorithm BESLRI, see previous paper.
- [2] Maximon, Leonard C., FORTRAN Program for Arbitrary Precision Arithmetic, unpublished data.
- [3] Olver, Frank W. J., Error Bounds for Asymptotic Expansion with an Application to Cylinder Functions of Large Argument, in *Asymptotic Solutions of Differential Equations and Their Applications*, Calvin H. Wilcox, Editor (Wiley & Sons, 1964).
- [4] National Bureau of Standards, *Handbook of Mathematical Functions*, Nat. Bur. Stand. (U.S.), Appl. Math. Ser. 55 (1964).
- [5] Certification of an algorithm for Bessel functions of complex argument. *J. Res. Nat. Bur. Stand. (U.S.)*, in press.

(Paper 77B3&4-386)