JOURNAL OF RESEARCH of the National Bureau of Standards — B. Mathematical Sciences
Vol. 77A, Nos. 3&4, July–December 1973

Bessel Functions of Real Argument and Integer Order

David J. Sookne [1]

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

$$I_n(x), \qquad n=0,1,\ldots,\mathrm{NB}-1$$

or

$$J_n(x), \qquad n=0,1,\ldots,\mathrm{NB}-1$$

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

$$y_{n-1}=\frac{2n}{x}y_n-\mathrm{SIGN}\cdot y_{n+1} \tag{1}$$

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

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

$$p_{n+1}=\frac{2n}{|x|}p_n-\mathrm{SIGN}\cdot p_{n-1} \qquad n=\mathrm{MAGX}+1,\ \mathrm{MAGX}+2,\ldots \tag{2}$$

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 $y_0^{(N)}, y_1^{(N)},\ldots$ is the recessive solution of (1) which satisfies the boundary condition $y_{\mathrm{MAGX}}=1$. From this solution, the I's and J's are found by normalizing

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

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

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

$$\mu = y_0^{(N)} + 2 \sum_{k=1}^{[N/2]} y_{2k}^{(N)} \qquad \text{for } J\text{'s}$$

$$\mu = \left(y_0^{(N)} + 2 \sum_{k=1}^{[N/2]} y_{2k}^{(N)} \right) \Big/ \cosh x \qquad \text{for } I\text{'s}. \tag{3}$$

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

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

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

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

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 $n = \text{MAGX}$, $\text{MAGX} - 1, \ldots, 1$ 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,

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

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

$$|y_{2k}^{(N)} - y_{2k}| = \left| \frac{1}{p_{2k+1}} \left\{ 1 + \frac{p_{2k}}{p_{2k+2}} + \frac{p_{2k}\ p_{2k+1}}{p_{2k+2}\ p_{2k+3}} + \ldots \right\} \right|$$

$$\leqslant \frac{1}{p_{2k+1}} \left\{ 1 + \frac{1}{\rho_{2k}^2} + \frac{1}{\rho_{2k}^4} + \ldots \right\} = \frac{\rho_{N'}^2}{(\rho_{N'}^2 - 1) p_{2k+1}};$$

compare section 2 above. Now let

$$R^{(N)} = \sum_{k=\left[\frac{N+1}{2}\right]}^{\infty} |y_{2k}|.$$

Then

$$R^{(N)} \leqslant \frac{\rho_{N'}^2}{\rho_{N'}^2 - 1} \sum_{k=\left[\frac{N+1}{2}\right]}^{\infty} \frac{1}{p_{2k+1}} \leqslant \frac{\rho_{N'}^2}{(\rho_{N'}^2 - 1) p_{N+1}} \left\{ 1 + \frac{1}{\rho_{N'}^2} + \frac{1}{\rho_{N'}^4} + \ldots \right\}$$

$$\leqslant \frac{\rho_{N'}^3}{(\rho_{N'}^2 - 1)^2 p_N}. \tag{7}$$

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

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

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

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

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

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

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

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

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

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

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

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

Then

$$|I_{\mathrm{MAGX}}(x)| = |J_{\mathrm{MAGX}}(ix)| = |J_{\mathrm{MAGX}}(\mathrm{MAGX}\cdot iz)|$$
$$\leqslant \left|\frac{z\exp\sqrt{1+z^2}}{1+\sqrt{1+z^2}}\right|^{\mathrm{MAGX}}.$$

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

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

Substituting this in (8), we obtain

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

Besides this approximate upper bound, the strict bounds

$$\left|\frac{R^{(N)}}{\mu}\right| < 7.6\cdot 10^{-\mathrm{NSIG}} \qquad |x| \geqslant 1$$

and

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

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

4. Appendix: Algorithm BESLRI

```
@ ELT BESLRI,1,730222, 58730

      SUBROUTINE BESLRI(X,NB,IZE,B,NCALC)
C THIS ROUTINE CALCULATES BESSEL FUNCTIONS I AND J OF REAL
C ARGUMENT AND INTEGER ORDER.
C
C
C      EXPLANATION OF VARIABLES IN THE CALLING SEQUENCE
C
C X       DOUBLE PRECISION REAL ARGUMENT FOR WHICH I*S OR J*S
C         ARE TO BE CALCULATED.  IF I*S ARE TO BE CALCULATED,
C         ABS(X) MUST BE LESS THAN EXPARG (WHICH SEE BELOW).
C NB      INTEGER TYPE.  1 + HIGHEST ORDER TO BE CALCULATED.
C         IT MUST BE POSITIVE.
C IZE     INTEGER TYPE.  ZERO IF J*S ARE TO BE CALCULATED, 1
C         IF I*S ARE TO BE CALCULATED.
C B       DOUBLE PRECISION VECTOR OF LENGTH NB, NEED NOT BE
C         INITIALIZED BY USER.  IF THE ROUTINE TERMINATES
C         NORMALLY (NCALC=NB), IT RETURNS J(OR I)-SUB-ZERO
C         THROUGH J(OR I)-SUB-NB-MINUS-ONE OF X IN THIS
C         VECTOR.
C NCALC   INTEGER TYPE, NEED NOT BE INITIALIZED BY USER.
C         BEFORE USING THE RESULTS, THE USER SHOULD CHECK THAT
C         NCALC=NB, I.E. ALL ORDERS HAVE BEEN CALCULATED TO
C         THE DESIRED ACCURACY.  SEE ERROR RETURNS BELOW.
C
C
C     EXPLANATION OF MACHINE-DEPENDENT CONSTANTS
C
C NSIG    DECIMAL SIGNIFICANCE DESIRED.  SHOULD BE SET TO
C         IFIX(ALOG10(2)*NBIT+1), WHERE NBIT IS THE NUMBER OF
C         BITS IN THE MANTISSA OF A DOUBLE PRECISION VARIABLE.
C         SETTING NSIG LOWER WILL RESULT IN DECREASED ACCURACY
C         WHILE SETTING NSIG HIGHER WILL INCREASE CPU TIME
C         WITHOUT INCREASING ACCURACY.  THE TRUNCATION ERROR
C         IS LIMITED TO T=.5*10**-NSIG FOR J*S OF ORDER LESS
C         THAN ARGUMENT, AND TO A RELATIVE ERROR OF T FOR
C         I*S AND THE OTHER J*S.
C NTEN    LARGEST INTEGER K SUCH THAT 10**K IS MACHINE-
C         REPRESENTABLE IN DOUBLE PRECISION.
C LARGEX  UPPER LIMIT ON THE MAGNITUDE OF X.  BEAR IN MIND
C         THAT IF ABS(X)=N, THEN AT LEAST N ITERATIONS OF THE
C         BACKWARD RECURSION WILL BE EXECUTED.
C EXPARG  LARGEST DOUBLE PRECISION ARGUMENT THAT THE LIBRARY
C         DEXP ROUTINE CAN HANDLE.
C
C
C                    ERROR RETURNS
C
C         LET G DENOTE EITHER I OR J.
C         IN CASE OF AN ERROR, NCALC.NE.NB, AND NOT ALL G*S
C  ARE CALCULATED TO THE DESIRED ACCURACY.
C         IF NCALC.LT.0, AN ARGUMENT IS OUT OF RANGE.  NB.LE.0
C  OR IZE IS NEITHER 0 NOR 1 OR IZE=1 AND ABS(X).GE.EXPARG.
C  IN THIS CASE, THE B-VECTOR IS NOT CALCULATED, AND NCALC
C  IS SET TO MIN0(NB,0)-1 SO NCALC.NE.NB.
```

```
C        NB.GT.NCALC.GT.0 WILL OCCUR IF NB.GT.MAGX AND ABS(G-
C   SUB-NB-OF-X/G-SUB-MAGX-OF-X).LT.10.**(NTEN/2), I. E. NB
C   IS MUCH GREATER THAN MAGX.  IN THIS CASE, B(N) IS CALCU-
C   LATED TO THE DESIRED ACCURACY FOR N.LE.NCALC, BUT FOR
C   NCALC.LT.N.LE.NB, PRECISION IS LOST.  IF N.GT.NCALC AND
C   ABS(B(NCALC)/B(N)).EQ.10**-K, THEN ONLY THE FIRST NSIG-K
C  SIGNIFICANT FIGURES OF B(N) MAY BE TRUSTED.  IF THE USER
C  WISHES TO CALCULATE B(N) TO HIGHER ACCURACY, HE SHOULD USE
C  AN ASYMPTOTIC FORMULA FOR LARGE ORDER.
C
      DOUBLE PRECISION
     1 X,B,P,TEST,TEMPA,TEMPB,TEMPC,EXPARG,SIGN,SUM,TOVER,
     2 PLAST,POLD,PSAVE,PSAVEL
      DIMENSION B(NB)
      DATA NSIG,NTEN,LARGEX,EXPARG/19,307,100000,7.D2/
      TEMPA=DABS(X)
      MAGX=IFIX(SNGL(TEMPA))
      IF(NB.GT.0.AND.MAGX.LE.LARGEX.AND.(IZE.EQ.0.OR.
     1(IZE.EQ.1.AND.TEMPA.LE.EXPARG))) GO TO 1
C ERROR RETURN -- X,NB,OR IZE IS OUT OF RANGE
      NCALC=MIN0(NB,0)-1
      RETURN
    1 SIGN=DBLE(FLOAT(1-2*IZE))
      NCALC=NB
C USE 2-TERM ASCENDING SERIES FOR SMALL X
      IF(TEMPA**4.LT..1D0**NSIG) GO TO 30
C INITIALIZE THE CALCULATION OF P*S
      NBMX=NB-MAGX
      N=MAGX+1
      PLAST=1.D0
      P=DBLE(FLOAT(2*N))/TEMPA
C CALCULATE GENERAL SIGNIFICANCE TEST
      TEST=2.D0*1.D1**NSIG
      IF(IZE.EQ.1.AND.2*MAGX.GT.5*NSIG) TEST=DSQRT(TEST*P)
      IF(IZE.EQ.1.AND.2*MAGX.LE.5*NSIG)TEST=TEST/1.585**MAGX
      M=0
      IF(NBMX.LT.3) GO TO 4
C CALCULATE P*S UNTIL N=NB-1.  CHECK FOR POSSIBLE OVERFLOW.
      TOVER=1.D1**(NTEN-NSIG)
      NSTART=MAGX+2
      NEND=NB-1
      DO 3 N=NSTART,NEND
        POLD=PLAST
        PLAST=P
        P=DBLE(FLOAT(2*N))*PLAST/TEMPA-SIGN*POLD
        IF(P-TOVER) 3,3,5
    3   CONTINUE
      N=NEND
C CALCULATE SPECIAL SIGNIFICANCE TEST FOR NBMX.GT.2.
      TEST=DMAX1(TEST,DSQRT(PLAST*1.D1**NSIG)*DSQRT(2.D0*P))
C CALCULATE P*S UNTIL SIGNIFICANCE TEST PASSES
    4 N=N+1
      POLD=PLAST
      PLAST=P
      P=DBLE(FLOAT(2*N))*PLAST/TEMPA-SIGN*POLD
      IF(P.LT.TEST) GO TO 4
      IF(IZE.EQ.1.OR.M.EQ.1) GO TO 12
C FOR J*S, A STRONG VARIANT OF THE TEST IS NECESSARY.
```

```
C CALCULATE IT, AND CALCULATE P*S UNTIL THIS TEST IS PASSED.
      M=1
      TEMPB=P/PLAST
      TEMPC=DBLE(FLOAT(N+1))/TEMPA
      IF(TEMPB+1.D0/TEMPB.GT.2.D0*TEMPC)TEMPB=TEMPC+DSQRT
     1(TEMPC**2-1.D0)
      TEST=TEST/DSQRT(TEMPB-1.D0/TEMPB)
      IF(P-TEST) 4,12,12
C TO AVOID OVERFLOW, DIVIDE P*S BY TOVER.  CALCULATE P*S
C UNTIL ABS(P).GT.1.
    5 TOVER=1.D1**NTEN
      P=P/TOVER
      PLAST=PLAST/TOVER
      PSAVE=P
      PSAVEL=PLAST
      NSTART=N+1
    6 N=N+1
      POLD=PLAST
      PLAST=P
      P=DBLE(FLOAT(2*N))*PLAST/TEMPA-SIGN*POLD
      IF(P.LE.1.D0) GO TO 6
      TEMPB=DBLE(FLOAT(2*N))/TEMPA
      IF(IZE.EQ.1) GO TO 8
      TEMPC=.5D0*TEMPB
      TEMPB=PLAST/POLD
      IF(TEMPB+1.D0/TEMPB.GT.2.D0*TEMPC)TEMPB=TEMPC+DSQRT
     1(TEMPC**2-1.D0)
C CALCULATE BACKWARD TEST, AND FIND NCALC, THE HIGHEST N
C SUCH THAT THE TEST IS PASSED.
    8 TEST=.5D0*POLD*PLAST*(1.D0-1.D0/TEMPB**2)/1.D1**NSIG
      P=PLAST*TOVER
      N=N-1
      NEND=MIN0(NB,N)
      DO 9 NCALC=NSTART,NEND
        POLD=PSAVEL
        PSAVEL=PSAVE
        PSAVE=DBLE(FLOAT(2*N))*PSAVEL/TEMPA-SIGN*POLD
        IF(PSAVE*PSAVEL-TEST) 9,9,10
    9   CONTINUE
      NCALC=NEND+1
   10 NCALC=NCALC-1
C THE SUM B(1)+2B(3)+2B(5)... IS USED TO NORMALIZE.  M, THE
C COEFFICIENT OF B(N), IS INITIALIZED TO 2 OR 0.
   12 N=N+1
      M=2*N-4*(N/2)
C INITIALIZE THE BACKWARD RECURSION AND THE NORMALIZATION
C SUM
      TEMPB=0.D0
      TEMPA=1.D0/P
      SUM=DBLE(FLOAT(M))*TEMPA
      NEND=N-NB
      IF(NEND) 17,15,13
C RECUR BACKWARD VIA DIFFERENCE EQUATION, CALCULATING (BUT
C NOT STORING) B(N), UNTIL N=NB.
   13 DO 14 L=1,NEND
        N=N-1
        TEMPC=TEMPB
        TEMPB=TEMPA
```

```
          TEMPA=(DBLE(FLOAT(2*N))*TEMPB)/X-SIGN*TEMPC
          M=2-M
    14    SUM=SUM+DBLE(FLOAT(M))*TEMPA
C STORE B(NB)
    15 B(N)=TEMPA
       IF(NB.GT.1) GO TO 16
C NB=1.  SINCE 2*TEMPA WAS ADDED TO THE SUM, TEMPA MUST BE
C SUBTRACTED
       SUM=SUM-TEMPA
       GO TO 23
C CALCULATE AND STORE B(NB-1)
    16 N=N-1
       B(N) =(DBLE(FLOAT(2*N))*TEMPA)/X-SIGN*TEMPB
       IF(N.EQ.1) GO TO 22
       M=2-M
       SUM=SUM+DBLE(FLOAT(M))*B(N)
       GO TO 19
C N.LT.NB, SO STORE B(N) AND SET HIGHER ORDERS TO ZERO
    17 B(N)=TEMPA
       NEND=-NEND
       DO 18 L=1,NEND
    18    B(N+L)=0.D0
    19 NEND=N-2
       IF(NEND.EQ.0) GO TO 21
C CALCULATE VIA DIFFERENCE EQUATION AND STORE B(N),
C UNTIL N=2
       DO 20 L=1,NEND
          N=N-1
          B(N)=(DBLE(FLOAT(2*N))*B(N+1))/X-SIGN*B(N+2)
          M=2-M
    20    SUM=SUM+DBLE(FLOAT(M))*B(N)
C CALCULATE B(1)
    21 B(1)=2.D0*B(2)/X-SIGN*B(3)
    22 SUM=SUM+B(1)
C NORMALIZE--IF IZE=1, DIVIDE SUM BY COSH(X).  DIVIDE ALL
C B(N) BY SUM.
    23 IF(IZE.EQ.0) GO TO 25
       TEMPA=DEXP(DABS(X))
       SUM=2.D0*SUM/(TEMPA+1.D0/TEMPA)
    25 DO 26 N=1,NB
    26    B(N)=B(N)/SUM
       RETURN
C TWO-TERM ASCENDING SERIES FOR SMALL X
    30 TEMPA=1.D0
       TEMPB=-.25D0*X*X*SIGN
       B(1)=1.D0+TEMPB
       IF(NB.EQ.1) GO TO 32
       DO 31 N=2,NB
          TEMPA=TEMPA*X/DBLE(FLOAT(2*N-2))
    31    B(N)=TEMPA*(1.D0+TEMPB/DBLE(FLOAT(N)))
    32 RETURN
       END
3.

END CUR LCC 1102-0038 L8
```

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)