Computer Physics Communications 47 (1987) 245–257
North-Holland, Amsterdam



# MODIFIED BESSEL FUNCTIONS $I_\nu(z)$ AND $K_\nu(z)$ OF REAL ORDER AND COMPLEX ARGUMENT, TO SELECTED ACCURACY

I.J. THOMPSON

*Department of Engineering Mathematics, University of Bristol, Bristol BS8 1TR, UK*

and

A.R. BARNETT

*Department of Physics, Manchester University, Manchester M13 9PL, UK*

Received 1 July 1987

## PROGRAM SUMMARY

*Title of program*: BESSCC

*Catalogue number*: ABBM

*Program obtainable from*: CPC Program Library, Queen's University, Belfast, N. Ireland (see application form in this issue)

*Computer*: NAS 7000 at Daresbury Laboratory. Also tested on CRAY-1, ATLAS-10, IBM 3081, CYBER 205, GEC 4190, CDC 7600, VAX 750, Zenith 248 and IBM PC/AT

*Operating system*: MVS

*Programming language used*: FORTRAN 77 with complex arithmetic in the required precision

*High speed storage required*: 360 kbytes, with 10520 bytes for BESSCC

*No. of bits in a real number*: 64

*Peripherals used*: reader, printer (neither used in BESSCC itself)

*No. of lines in combined program and test deck*: 527 (1/3 are comments)

*Keywords*: general purpose, Bessel, continued fraction, Temme, cylindrical, Miller's algorithm, Steed's method, backward recurrence, Airy, Kelvin, modified

*Nature of physical problem*
The BESSCC subroutine calculates the modified Bessel functions $I_\nu(z)$ and $K_\nu(z)$ (and derivatives) for complex argument $z$ and a sequence of real orders $\nu$, $\nu+1,\ldots,\nu+N-1$ for integer $N \geq 1$. These functions arise in the solutions of potential problems in spherical and cylindrical coordinates. They can also be used to calculate ordinary Bessels $J_\nu(z)$, $Y_\nu(z)$, spherical Bessels $j_\nu(z)$, $y_\nu(z)$, Kelvin and Airy functions.

*Method of solution*
For large arguments $z$, Temme's algorithm [1] is used to find $K_\nu$, $K'_\nu$ and $I_\nu$, $I'_\nu$. The $I_\nu(z)$ values are recurred upward (if this is stable). For moderate $z$, $K_\nu$ and $K'_\nu$ are found using Temme's method, and Miller's method is used to find $I'_\nu/I_\nu$, with the $I_\nu$ normalised by the Wronskian with $K_\nu$. For small $z$, Miller's method is again used for the $I_\nu$, and a Neumann series for the $K_\nu(z)$.

Upward recurrence of the $K_\nu$ is always stable, and downward recurrence for the $I_\nu$ is used in the second and third cases.

*Restrictions on the complexity of the problem*
The functions are determined only for real order $\nu > -\frac{1}{2}$. Reflection formulae are given for $\nu \leq -\frac{1}{2}$, and for complex order $\nu$ the procedure COULCC of ref. [2] is available. The routines are less efficient when both order and argument are large, becoming noticeable when $\nu + N > |z|/2 > 1000$.

*Typical running time*
The test deck takes 0.44 s of execution time on a NAS 7000.

*Accuracy*
In general, results within two digits of machine accuracy may be obtained, subject to the correct set of constants being included. The code is released with constants allowing accuracies of up to 24 significant digits. For calculations of arbitrary precision and/or of more than 24 digits, an appended

0010-4655/87/$03.50 © Elsevier Science Publishers B.V.
(North-Holland Physics Publishing Division)

program ZETA can be easily used to calculate the required constants.

On IBM machines, BESSCC has been compared with the similar package of Campbell [3] and with the larger suite of programs of Amos [4], and no discrepancies are found greater than $10^{-14}$. A table is given of the accuracies obtainable with different machines.

*References*

[1] N.M. Temme, Numer. Math. 41 (1983) 63; J. Comput. Phys. 19 (1975) 324.

[2] I.J. Thompson and A.R. Barnett, Comput. Phys. Commun. 36 (1985) 363.

[3] J.B. Campbell, Comput. Phys. Commun. 24 (1981) 97.

[4] D.E. Amos, Trans. on Num. Soft 12 (1986) 265.

## LONG WRITE-UP

### 1. Introduction

Algorithms are presented for the computation of the modified Bessel functions $I_\nu(z)$ and $K_\nu(z)$, of real order $\nu$, and complex argument $z$. From these, other Bessel functions can be computed, namely $J_\nu$, $Y_\nu$, $j_\nu$, $y_\nu$, the Airy functions and the Kelvin functions (section 2.6). The accompanying subroutine BESSCC is written in FORTRAN 77, and for the required relative accuracy calculates $I_\nu(z)$, $I'_\nu(z)$, $K_\nu(z)$ and $K'_\nu(z)$ for a specified sequence of orders $\nu$, $\nu+1,\ldots,\nu+N-1$. An integer variable IFAIL returns the number of orders which could not be calculated because overflow or underflow would occur. Optional exponential scaling may be employed to reduce the likelihood of this error occurring.

Stored constants in the program are given to 24 digits of precision, and give the continued fraction expansions for the functions $g_1(t)$ and $g_2(\nu)$ (see section 3.3). The number of these terms required is proportional to the required number of digits of precision, over the range $|\nu| \le \frac{1}{2}$. Should greater precision be needed, the small program ZETA (appended to the accompanying CPC deck) can be used. This program calculates the BESSCC constants using a working precision approximately 6 digits greater than the target Bessel precision, and thus enables Bessel calculations of arbitrary accuracy subject to the machine arithmetic being available. The ZETA routines could have been called directly from BESSCC, but for the sake of the speed of the Bessel calculations they were separated in the above manner.

The subroutine BESSCC (with the associated function CF2E) comprises 473 lines of code and requires 10520 bytes of storage. The remainder of the deck consists of a test program, and the program ZETA.

### 2. Bessel function expansions

#### 2.1. Temme's algorithm for the irregular solutions

As explained in refs. [1,2], the sequence of values

$$P_k = {}_2F_0(a+k,\ b+k;;\ u)$$

of the confluent hypergeometric function satisfy both the recurrence relations

$$P_k = \left[1-(a_k+b_k+1)u\right]P_{k+1} - u^2 a_k b_k P_{k+2},$$

where $a_k = a+k$ and $b_k = b+k$, and Temme's sum rule

$$\sum_{k=0}^{\infty} C_k P_k = 1 \qquad (2.1)$$

where

$$C_0 = 1 \text{ and } C_{k+1} = -a_k b_k u C_k/(k+1).$$

These sums are convergent when $a > 0$, $0 < b < 1$, and $\mathrm{Re}(u) < 0$ with $|u| \ll 1$, so Miller's backward recurrence method can be used to calculate a normalised sequence of values $P_k$ for $k$ from some starting order $M$ down to 0. We use an extension of Steed's method of section 3.1, which finds the upper order $M$ automatically. As in ref. [3], the ratio $abP_1/P_0$ is the logarithmic derivative of ${}_2F_0(a,\ b;;\ u)$ with respect to $u$, and this gives [2] the continued fraction $\mathrm{CF2}^\sigma$ for $\sigma = \pm 1$ by

$$\mathrm{CF2}^\sigma = \mathrm{i}\sigma\left[1 + 2abuP_1^\sigma/P_0^\sigma\right], \qquad (2.2)$$

where [4] the ratio $P_1/P_0$ converges to a given accuracy with approximately half as many terms as needed for the sum (2.1).

The function $P_k^\sigma$ for $a = \frac{1}{2}-\nu$, $b = \frac{1}{2}+\nu$, and $u = -\sigma/(2z)$ can be found for both $\sigma = +1$ and

$-1$, but when $\mathrm{Re}(z) > 0$ the $\sigma = +1$ Temme sum converges more quickly, giving

$$K_\nu = \tfrac{1}{2}\pi(\pi/(2z))^{1/2} e^{-z} P_0^+ \tag{2.3}$$

and

$$K_\nu'/K_\nu = \mathrm{iCF2}^+ - 1/(2z). \tag{2.4}$$

When $|z| \gg 10$, the functions $P_k^-$ also converge by this method. We need however only find the ratio $P_1^-/P_0^-$, if we already have the value of $P_k^+$, because of the Wronskian relation $P_0^+ P_0^- [\mathrm{CF2}^+ - \mathrm{CF2}^-] = 2\mathrm{i}$. Then, guided by our Coulomb wavefunction results [2,5] as to where we can calculate the regular solution $F$ (equivalent to $I$) as the difference of two irregular solutions $H^+$ and $H^-$, we have the new results:

$$I_\nu(z) = \tfrac{1}{2}(\pi/(2z))^{1/2}\, e^z P_0^- [1 - H^+/H^-], \text{ and}$$

$$I_\nu'/I_\nu = \mathrm{i}[\mathrm{CF2}^- - (H^+/H^-)\mathrm{CF2}^+]/[1 - H^+/H^-] - 1/(2z),$$

where the ratio $H^+/H^-$ of the Coulomb functions is

$$H^+/H^- = \exp[-2z + (\nu - \tfrac{1}{2})\pi\mathrm{i}]\, P_0^+/P_0^-.$$

### 2.2. Recurrence relations, and Miller's method for $I_\nu'/I_\nu$

The functions $I_\nu$ and $\exp(\nu\pi\mathrm{i})K_\nu$ both satisfy the same recurrence relations [6, eqs. 9.6.26]:

$$I_\nu(z) = I_{\nu+1}'(z) + ((\nu+1)/z) I_{\nu+1}(z)$$

and

$$I_\nu'(z) = (\nu/z) I_\nu(z) + I_{\nu+1}(z),$$

with equivalent relations for upward recurrences. These are stable provided that the functions do not monotonically decrease in the direction of recurrence. The stable directions are downward recurrence of the $I_\nu(z)$ throughout the complex $z$ plane, and upward recurrence of $K_\nu(z)$ in the right-hand half plane $\mathrm{Re}(z) \geq 0$. Furthermore, when $|z|$ is large, the values of $I_\nu(z)$ decrease only very slowly as $\nu$ increases from near zero until approximately as far as $0.6|z|$. For large $|z|$ this enables BESSCC (in contrast to the methods of refs. [1,8]) to recur the $I_\nu$ upward for some distance with very little loss of accuracy, provided the moduli $|I_\nu|$ are monitored to avoid the errors rising by more than, say, one order of magnitude.

From the downward recurrence relations for the $I_\nu$ we have Miller's backward recurrence method, as extended by Gautschi [9] to iterate on $f_\nu = I_\nu'/I_\nu$. For a starting order $M$ the method uses

$$f_{\nu+k-1} = (\nu+k-1)/z + 1/[(\nu+k)/z + f_{\nu+k}]$$

for $k$ from $M$ down to 1, beginning with (for example)

$$f_{\nu+M} = (\nu+M)/z + z/[2(\nu+M+1)].$$

We adopt Sookne's method [10] to find the starting order $M$ by performing a preliminary upward recurrence of $I_j$ from $J = \max(\nu, |z|)$ and $I_{J-1} = 0$, $I_J = 1$, using the equation

$$I_{j+1} = (2j/z) I_j - I_{j-1}$$

until a modulus $|I_{\nu+M}|$ is found that is greater than $\epsilon^{-1/2}$ for a relative accuracy $\epsilon$ in the resulting ratio $f_\nu$. Although this method requires more work than the use of pre-computed starting orders (as is done in refs. [7,8]), it does enable calculations to attain any required precision. We found Sookne's method to be quicker using the extended Steed's method of section 3.1 since less work is involved at each step.

### 2.3. Small-z sum rules for $I_\nu$, and Neumann series for $K_\nu$

For $\mathrm{Re}(z) < 2$, the normalising series

$$\sum_{m=0}^{\infty} u_m I_{\nu+2m}(z) = 1 \tag{2.5}$$

can be used, with

$$u_m = (-1)^m (2/z)^\nu (\nu+2m)\Gamma(\nu+m)/m!,$$

that is, with

$$u_0 = (2/z)^\nu \Gamma(1+\nu)$$

and

$$u_m = -u_{m-1}[(\nu+2m)/(\nu+2m-2)](\nu+m)/m.$$

The same $I_{\nu+2m}(z)$ values can be used in a Neumann series to give the irregular function $K_\nu(z)$, following Goldstein and Thaler [11], and Campbell [7] but with simplifications that arise because BESSCC calculates the $I$ and $K$ Bessels together and preserves common terms:

$$K_\nu(z) = d_0 I_\nu(z) + \sum_{m=1}^{\infty} (\nu+2m) D_m I_{\nu+2m}(z), \tag{2.6}$$

with

$$d_0 = g_1(\pi z) + u_0 \sinh[\nu(g_2(\nu) + \ln(2/z))]/\mu,$$
$$g_1(t) = \tfrac{1}{2}\pi[t^{-1} - \operatorname{cosec}(t)],$$
$$g_2(\nu) = \nu^{-1} \ln \Gamma(1+\nu),$$
$$D_1 = u_0^2(\nu+2)/(1-\nu)$$

and

$$D_m = D_{m-1}(2\nu+m-1)(\nu+m-1)/(m(m-\nu))$$
for $m \geq 2$.

### 2.4. The Wronskian relation

At each order $\nu$, the functions satisfy the relation

$$I'_\nu K_\nu - K'_\nu I_\nu = 1/z, \tag{2.7}$$

and this is used in BESSCC to calculate $K'_\nu$ from $I_\nu$, $I'_\nu$ and $K_\nu$. This is provided that $I_\nu$ is not near a zero, as otherwise there would be significant cancellation errors: in such cases BESSCC evaluates $K'_\nu$ from $K_\nu$ and $K'_\nu/K_\nu$ via the continued fraction CF2$^+$ that is already coded as in section 2.1.

### 2.5. Analytic continuations for Re(z) < 0 and $\nu < 0$

The analytic continuation for $\mathrm{Re}(z) < 0$ is given by eqs. 9.6.30 and 31 of ref. [6]:

$$I_\nu(tz) = t^\nu I_\nu(z) \quad \text{and}$$
$$K_\nu(tz) = t^{-\nu} K_\nu(z) - \pi\mathrm{i}\ \mathrm{Im}(t^\nu)\ \operatorname{cosec}(\nu\pi) I_\nu(z)$$

where $t = \exp(\pm\pi\mathrm{i})$. These transformations are included in the code.

For negative $\nu$ one can use

$$I_{-\nu}(z) = I_\nu(z) + (2/\pi)\sin(\nu\pi) K_\nu(z) \quad \text{and}$$
$$K_{-\nu}(z) = K_\nu(z).$$

### 2.6. Other Bessel functions in terms of $I_\nu(z)$ and $K_\nu(z)$

The modified functions can be used as a basis to express all the related solutions of the various forms of Bessel's equation. Abramowitz and Stegun [6] list the relevant formulae in chapters 9, 10.

#### 2.6.1. Cylindrical Bessel functions $J_\nu(z)$ and $Y_\nu(z)$

By inverting eq. (9.6.3), p. 375 of ref. [6] there follows, using for the 90° rotation $p = \exp(\mathrm{i}\pi/2)$,

$$J_\nu(z) = pI_\nu(zp^{-1}) \qquad \text{for } -\pi/2 < \arg z \leq \pi,$$
$$= p^3 I_\nu(zp^3) \qquad \text{for } -\pi < \arg z \leq \pi/2,$$

with the plane being cut along the negative $z$-axis. Identical expressions follow for $Y_\nu(z)$ in terms of the irregular solution $K_\nu(z)$. The derivatives of $J$ and $Y$ are readily found from $I'_\nu$ and $K'_\nu$. (Problems which arose in the evaluation of the Coulomb functions [2,5] are absent here.)

#### 2.6.2. Spherical Bessel functions $j_\nu(z)$ and $y_\nu(z)$

The defining relation [6, 10.1.1] is in terms of the $J$, $Y$ functions:

$$j_\nu(z) = \sqrt{(\tfrac{1}{2}\pi/z)}\, J_{\nu+1/2}(z),$$
$$y_\nu(z) = \sqrt{(\tfrac{1}{2}\pi/z)}\, Y_{\nu+1/2}(z),$$

where $\sqrt{z}$ has a cut along the negative real axis. This route to the spherical Bessels can be followed for real values of $\nu$ and complex $z$ values, but if integer orders with complex arguments are required then a simpler method is presented in fig. 1, which gives the self-contained subroutine SBESJH. For integer $n \geq 0$ it computes the regular solution $j_n(z)$ and the irregular Hankel solution $h_n(z)$ (and their derivatives with respect to $z$) in the half-plane $\mathrm{Im}(z) > -3$ (where $h_0^{(1)}(z)$ is exponentially decaying). The remaining spherical

```
      SUBROUTINE SBESJH (X,LMAX,XJ,XJP,XH1,XH1P,IFAIL)
C ***                                                        I.J.Thompson
C ***                                                        31 May 1985.
C ***  COMPLEX SPHERICAL BESSEL FUNCTIONS from l=0 to l=LMAX
C ***    for X in the UPPER HALF PLANE ( Im(X) > -3)
C ***
C ***    XJ(l)   = j/l(x)          regular solution: XJ(0)=sin(x)/x
C ***    XJP(l)  = d/dx j/l(x)
C ***    XH1(l)  = h(1)/l(x)       irregular Hankel function:
C ***    XH1P(l) = d/dx h(1)/l(x)            XH1(0) = j0(x) + i. y0(x)
C ***                                               =(sin(x)-i.cos(x))/x
C ***                                               = -i.exp(i.x)/x
C ***   IFAIL = -1 for arguments out of range
C ***         = 0  for all results satisfactory
C ***         > 0  for results ok up to & including order LMAX-IFAIL
C ***
C ***  Using complex CF1, and trigonometric forms for l=0 solutions.
C ***  Note real routine in CPC 21 (1981) 297
C ***
      IMPLICIT COMPLEX*16 (A-H,O-Z)
      PARAMETER (LIMIT=20000)
      DIMENSION XJ(0:LMAX),XJP(0:LMAX),XH1(0:LMAX),XH1P(0:LMAX)
      REAL*8 ZERO,ONE,ACCUR,TM30,ABSC
      DATA ZERO,ONE/ 0.0D0,1.0D0 /, ACCUR /1.0D-12/, TM30 / 1D-30 /,
     #     CI / (0D0,1D0) /
      ABSC(W) = ABS(REAL(W)) + ABS(IMAG(W))
      IFAIL= -1
      IF(ABSC(X).LT.ACCUR .OR. IMAG(X).LT.-3.0) GO TO 5
      XI = ONE/X
      W  = XI + XI
      PL = LMAX*XI
      F = PL + XI
      B  = F + F + XI
      D  = ZERO
      C  = F
      DO 1 L=1,LIMIT
      D  = B - D
      C  = B - ONE/C
         IF(ABSC(D).LT.TM30) D = TM30
         IF(ABSC(C).LT.TM30) C = TM30
      D = ONE / D
      DEL= D * C
      F = F * DEL
      B = B + W
    1 IF(ABSC(DEL-ONE).LT.ACCUR) GO TO 2
        IFAIL = -2
        GO TO 5
C
    2 XJ(LMAX)   = TM30
      XJP(LMAX)  = F * XJ(LMAX)
C
C *** Downward recursion to l=0 (N.B.  Coulomb Functions)
C
      DO 3 L = LMAX-1,0,-1
      XJ(L) = PL*XJ(L+1) + XJP(L+1)
      XJP(L)= PL*XJ(L)   - XJ(L+1)
    3 PL = PL - XI
C *** Calculate the l=0 Bessel Functions
      XJ0  = XI * SIN(X)
      XH1(0) = EXP(CI*X) * XI * (-CI)
      XH1P(0)= XH1(0) * (CI - XI)
C
C *** Rescale XJ, XJP,  converting to spherical Bessels.
C *** Recur   XH1,XH1P             AS spherical Bessels.
C
        W = ONE/XJ(0)
        PL = XI
      DO 4  L = 0,LMAX
      XJ(L)  =  XJ0*(W*XJ(L))
      XJP(L) =  XJ0*(W*XJP(L)) - XI*XJ(L)
         IF(L.EQ.0) GO TO 4
      XH1(L) = (PL-XI) * XH1(L-1) - XH1P(L-1)
         PL = PL + XI
      XH1P(L)=- PL     * XH1(L)   + XH1(L-1)
    4 CONTINUE
      IFAIL = 0
      RETURN
    5    WRITE(6,10) IFAIL
   10    FORMAT( 'SBESJH : IFAIL = ',I4)
      RETURN
      END
```

Fig. 1. A separate routine for spherical Bessels of complex argument and integer order.

Bessels may be found by

$$y_n(z) = \mathrm{i}\left[ j_n(z) - h_n^{(1)}(z)\right],$$

$$h_n^{(2)}(z) = h_n^{(1)}(z) - 2\mathrm{i}\, y_n(z).$$

### 2.6.3. *Airy functions Ai(z) and Bi(z)*

These functions are combinations of Bessel functions of order 1/3 [6, section 10.4], namely

$$\mathrm{Ai}(z) = (1/\pi)\sqrt{(z/3)}\, K_{1/3}(\zeta),$$

$$\mathrm{Bi}(z) = \left(\sqrt{z}/\pi\right)\left[\left(2\pi/\sqrt{3}\right) I_{1/3}(\zeta) + K_{1/3}(\zeta)\right],$$

where $\zeta = (2/3)z^{3/2}$. Near the origin it becomes more sensible to program eq. 10.4.14 [6] for $\mathrm{Ai}(z)$ directly as the difference between the $I_{-1/3}$ and $I_{1/3}$ functions, using [6, 9.6.10] since the singularities at $z = 0$ can be controlled.

### 2.6.4. *Kelvin functions ker(x), kei(x), ber(x), bei(x)*

For practical purposes these are functions of the real argument $x$ lying between 0 and about 10. The various real Kelvin functions are the real and imaginary parts of $I_{0,1}$ and $K_{0,1}$ functions of argument $\pm x\mathrm{i}^{1/2}$:

$$\mathrm{ker}(x) + \mathrm{i}\ \mathrm{kei}(x) = K_0(x\mathrm{i}^{1/2}),$$

$$\mathrm{ber}(x) + \mathrm{i}\ \mathrm{bei}(x) = I_0(x\mathrm{i}^{1/2}),$$

$$\mathrm{ker}_1(x) + \mathrm{i}\ \mathrm{kei}_1(x) = \mathrm{i}^{-1}K_1(x\mathrm{i}^{1/2}),$$

$$\mathrm{ber}_1(x) + \mathrm{i}\ \mathrm{bei}_1(x) = \mathrm{i}I_1(-x\mathrm{i}^{1/2}).$$

Kelvin functions of order $\nu$ relate to the same order modified Bessel functions together with a phase factor of $\mathrm{i}^{\pm\nu}$.

## 3. Numerical methods

### 3.1. *Forward evaluation of continued fractions and Temme's sum*

Given a recurrence relation $P_{k-1} = b_k P_k + a_{k+1}P_{k+1}$ such as that in section 2.1, Steed's algorithm [3] can be used to evaluate the continued fraction

$$P_1/P_0 = 1/\left(b_1 + a_2/\left(b_2 + a_3/\cdots\right)\right)$$

forwards from $k=0$ as the successive sum $P_1/P_0 = \Sigma h_k$, a method not requiring a precalculated starting order which varies according to the accuracy desired. The method has been extended to calculate a forward approximation to the sum required in Temme's method,

$$S_N = \sum_{k=1}^{N} C_k P_k / P_0$$

as

$$S_N = \sum_{i=1}^{N} Q_i h_i$$

where

$$Q_i = \sum_{k=1}^{i} C_k q_k$$

and $q_k$ is the forward sequence $q_k = (q_{k-1} - b_k q_k)/a_{k+1}$ starting from $q_0 = 0$ and $q_1 = 1$.

### 3.2. Calculation cases by (z, ν) region

There are three 'cases' for combining the various expansions of section 2. First, the argument $z$ is reflected if necessary into the right-hand half plane $\mathrm{Re}(z) \geq 0$, and a $\nu_{\min}$ is found with $|\nu_{\min}| \leq \frac{1}{2}$ such that $\nu - \nu_{\min}$ is integral. The following rules are chosen to minimise boundary errors for selections of $\nu_{\min}$ values in the range $-0.5$ to 0.5.

For large arguments, $|z| > 25$, and $\nu_{\max} < 0.6|z|$, case 1 uses Temme's algorithm for both the $P_k^+$ and $P_k^-$ sequence at $\nu_{\min}$, to find $K_\nu$, $K_\nu'$ and $I_\nu$, $I_\nu'$. The $I_\nu(z)$ values are recurred upward while this is stable. Should instabilities be detected, the remaining values are calculated by the slower method of case 2.

Case 2 is for moderate $|z| > 3$ and $10\mathrm{Re}(z) + \mathrm{Im}(z) \geq 20$, when the values $K_\nu$ and $K_\nu'$ are found using Temme's algorithm at $\nu_{\min}$ for the $P_k^+$ as in case 1, and Miller's method is used to find logarithmic derivative $I_\nu'/I_\nu$ at $\nu_{\max}$. The $I_\nu$ are normalised using the Wronskian with the $K_\nu$ at $\nu_{\min}$.

For small $|z| < 3$, or $10\mathrm{Re}(z) + \mathrm{Im}(z) < 20$, case 3 is used, with Miller's method and the sum rule of eq. (2.5) for the $I_\nu$ and $I_\nu'$, and a Neumann series for the $K_\nu(z)$ at $\nu_{\min}$. The value of $K_\nu'$ at $\nu_{\min}$ is found by the Wronskian, unless $I_\nu$ is near a zero, in which case CF2 is called again.

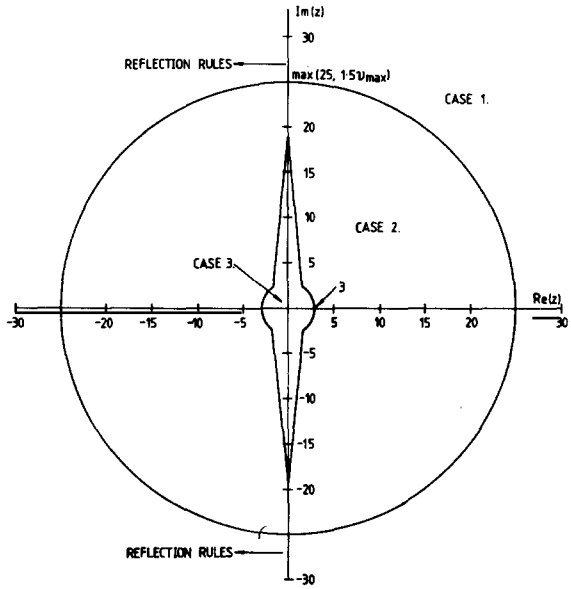


Fig. 2. Parameter cases for BESSCC calculations.

Upward recurrence of the $K_\nu$ and $K_\nu'$ is always stable, and downward recurrence for the $I_\nu$ and $I_\nu'$ is used in the second and third cases. The effects of these rules for calculations in the complex $z$ plane are shown in fig. 2, where it is assumed that the upward recurrence of the $I_\nu$ in case 1 proves to be stable.

### 3.3. Continued fractions for $g_1(t)$ and $g_2(\nu)$: the ZETA program

The functions

$$g_1(t) = \tfrac{1}{2}\pi\left[t^{-1} - \mathrm{cosec}(t)\right] \quad \text{for } t = \pi\nu$$

and

$$g_2(\nu) = \nu^{-1} \ln \Gamma(1+\nu)$$

do not have closed form expressions over the range $|\nu| \leq \frac{1}{2}$. For that reason we start with their series expansions

$$g_1(t) = \tfrac{1}{2}\pi \sum_{n=0}^{\infty} (-1)^n 2(2^{2n-1} - 1) B_{2n} t^{2n-1}/(2n)!$$

and

$$g_2(\nu) = -\gamma - \sum_{n=1}^{\infty} (-\nu)^n \zeta_{n+1}/(n+1)$$

where $B_{2n}$ are the Bernoulli numbers [6, table 23.2],

$$\gamma = -\psi^{(0)}(1) = 0.57721...$$

is Euler's constant, and

$$\zeta_{n+1} = (-1)^{n+1}\psi^{(n)}(1)/n! = \sum_{k=1}^{\infty} k^{-n-1}$$

is the Riemann zeta function for integer arguments [6, table 23.3].

To calculate these functions to arbitrary accuracy, we use the fact that the Bernoulli numbers have exact rational values, and that the polygamma function $\psi^{(n)}(z)$ has an asymptotic expansion in terms of the same Bernoulli numbers. We have therefore adapted a digamma ($\psi^{(0)}$) code of Kölbig [12] to calculate polygamma functions of arbitrary order $n$, in order to calculate the series coefficients to the accuracies required.

Having the series expansion for the required function, we have calculated the coefficients of the corresponding continued fraction using the transformations and code DFRACT of ref. [13]. The continued fraction coefficients are calculated by the program ZETA, and inserted in BESSCC as DATA statements. From fig. 3 we see that the number of terms required is a linear function of the number of digits of accuracy required, and the equations giving the straight lines in the figure are built into the BESSCC code.

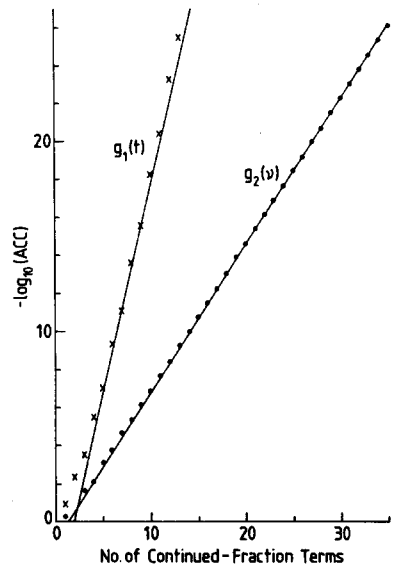


Fig. 3. Number of terms required vs. number of digits of accuracy required.

## 4. Program description

### 4.1. The BESSCC calling sequence

The calling sequence for the program is (see fig. 4)

CALL BESSCC(Z, XNU, NL, FI, FK, FIP,
FKP, MODE, ACC, IFAIL)

where the arguments have the following type and meaning:

| | | |
|---|---|---|
| Z | complex argument | $z$, non-zero |
| XNU | real | minimum order $\nu \geq -\frac{1}{2}$ |
| NL | integer | number of orders $\nu$, $\nu+1, \ldots, \nu+\mathrm{NL}-1$ required |
| FI | complex array | dimension NL, regular Bessel $I_\nu(z)$ |
| FK | complex array | dimension NL, irregular Bessel $K_\nu(z)$ |
| FIP | complex array | dimension NL, regular derivative $I'_\nu(z)$ |
| FKP | complex array | dimension NL, irregular derivative $K'_\nu(z)$ |
| MODE | integer | \|MODE\| gives the selection of $I$, $K$, $I'$ and $K'$, and MODE < 0 selects exponential scaling: |
| if \|MODE\| | | |
| = 1 | $I$, $K$, $I'$, $K'$ | functions are computed and stored. |

$= 2$ $I$, $K$
$= 3$ $I$, $I'$
$= 4$ $I$ only.

if MODE $< 0$ then the values returned are scaled by an exponential factor (dependent only on $z$) to bring nearer unity the functions for large $|z|$ and small $|\nu| < |z|$:
so FI = exp(− |Re($z$)|) * $I$
FIP = exp(− |Re($z$)|) * $I'$
and FK = exp(Re($z$)) * $K$
FKP = exp(Re($z$)) * $K'$

| | | |
|---|---|---|
| ACC | real | target relative accuracy<br>If ACC > 0.0001,<br>or the code finds that 1.0 + ACC = 1.0,<br>then a default ACCDEF = $10^{-6}$ is used. |
| IFAIL | integer | classification of errors on output: |
| IFAIL in output: | − 2 = | argument out of range |
| | − 1 = | one of the continued fractions failed,<br>or arithmetic check before final recursion |
| | 0 = | All calculations satisfactory |
| | > 0: | results available for orders below<br>position NL-IFAIL in the output arrays. |

Further information about performance of BESSCC is provided by a named common block /BSTEED/ containing in order ACCUR (REAL * 8) and the integers NFP, NPQ(2) and KASE.
ACCUR = adopted target accuracy for the calculation, normally ACC.
NFP = starting order $M$ for Miller's method for regular solution
NPQ(1) = number of terms required for Temme's sum for $P^+$
NPQ(2) = number of terms required for CF2$^-$, and
KASE = 1, 2 or 3 according the case of section 3.2.

### 4.2. Test deck

The test deck contains a main program to call BESSCC for a succession of $z$, $\nu$ and NL combinations determined by the data read in. Prior to calling BESSCC, the main program finds the smallest ACC such that 1.0 + ACC = 1.0, and it prints its value to remind the user of his machine's maximum precision. BESSCC requests this target accuracy in the Bessel calculations, though experience suggests that the achieved accuracy will be bounded by, at best, 50 times the ACC limit.

### 4.3. Operation of the ZETA program

The ZETA program included at the end of the CPC deck is a stand-alone program that calculates the continued fraction coefficients for the functions $g_1(t)$ and $g_2(\nu)$ as described in section 3.3. In order to obtain 24 digits of accuracy for these coefficients, the program is written to use the quadruple-precision complex arithmetic (COMPLEX * 32) that is available with the IBM VS-FORTRAN compiler.

The inputs to the program are (in free format) the real numbers ACCUR and ACC, where ACCUR is the relative precision available for the ZETA program to use (1Q − 33 with VS FORTRAN as above), and ACC is the maximum target accuracy that will be used in the BESSCC program (e.g. 1Q − 24). The program then outputs on a 'punch' file 7 the DATA cards needed for

```
      SUBROUTINE BESSCC(ZZ,XNU,NL, FI,FK,FIP,FKP, MODE1,ACC,IFAIL)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C  COMPLEX I,K BESSEL FUNCTIONS PROGRAM USING STEED'S METHOD           C
C                                                                      C
C  I. J. Thompson          Bristol     JULY    1986                   C
C                                                                      C
C  original program  RCWFN       in    CPC  8 (1974) 377-395         C
C                  +  RCWFF       in    CPC 11 (1976) 141-142         C
C                  +  COULFG      in    CPC 27 (1982) 147-166         C
C                  +  COULCC      in    CPC 36 (1985) 363-372         C
C  description of real algorithm in    CPC 21 (1981) 297-314         C
C  description of complex algorithm    JCP 64 (1986) 490-509         C
C  this version written up         in   CPC ..                       C
C                                                                      C
C  BESSCC returns I,K,I',K' for complex Z, real XNU, & integer NL > 0  C
C   for integer-spaced orders  XNU  to  XMU = XNU + NL - 1             C
C  The first order XNU must be > -0.5                                  C
C                                                                      C
C  if |MODE1|= 1  get I,K,I',K'   for integer-spaced NU values         C
C            = 2       I,K       unused arrays must be dimensioned in C
C            = 3       I,  I'          call to at least length (1)   C
C            = 4       I                                             C
C                                                                      C
C     if MODE1<0 then the values returned are scaled by an exponential C
C                factor (dependent only on Z) to bring nearer unity    C
C                the functions for large |Z|, small |XN| < |Z|         C
C     So define SCALE = (  0          if MODE1 > 0                     C
C                       (  REAL(Z)    if MODE1 < 0                     C
C        then FI = EXP(-ABS(SCALE)) * I                                C
C             FIP= EXP(-ABS(SCALE)) * I'                               C
C         and FK = EXP(SCALE)       * K                              C
C             FKP= EXP(SCALE)       * K'                             C
C                                                                      C
C  Precision:  results to within 1-2 decimals of 'machine accuracy',   C
C              depending on the value of ACC in the calling sequence.  C
C   If ACC is too small or too large, a default ACCDEF is used         C
C                                                                      C
C   BESSCC is coded for REAL*8 on IBM or equivalent  ACC > 2D-16       C
C                                                                      C
C   Use IMPLICIT COMPLEX*32 & REAL*16 on VS compiler   ACC > 1Q-24     C
C     (More GAM & CSC coefficients can be provided for ACC = 1Q-31)    C
C   For single precision CDC, CRAY etc reassign REAL*8=REAL etc.       C
C                                                                      C
C   IFAIL  in output:  -2 = argument out of range                      C
C                      -1 = one of the continued fractions failed,     C
C                           or arithmetic check before final recursion C
C                       0 = All Calculations satisfactory              C
C                      ge 0 : results available for orders up to & at  C
C                             position NL-IFAIL in the output arrays.  C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Fig. 4. Subroutine BESSCC, with introductory comment cards.

```
C   DATA statements for target accuracies up to 1.0D-14
      DATA GAM /
     X -5.7721566490153286D-01,-5.7721566490153286D-01,
     X  1.4248868896592017D+00,-9.3771157672490940D-01,
     X -9.7734772729488805D-02, 6.1306158583576872D-01,
     X  1.0355483535978465D-01, 3.8334536763573750D-01,
     X  1.6760985331801693D-01, 3.4231530708797397D-01,
     X  1.8720963846648549D-01, 3.0441764520575319D-01,
     X  2.0380577436752762D-01, 3.0369222399529579D-01,
     X  2.1114734365534483D-01, 2.8154925329861021D-01,
     X  2.1866203993615180-01, 2.8875999269033998D-01,
     X  2.2113656863567650D-01, 2.7119312296069024D-01,
     X  2.2785164017712757D-01/
      DATA CSC, PI2 /
     X  1.6666666666666667D-01,-1.1666666666666667D-01,
     X  1.1224489795918367D-02,-2.8396206967635539D-02,
     X  3.8818310143174027D-03,-1.2566901682859146D-02,
     X  1.9590158400391263D-03,-7.0586744483313501D-03,
     X  1.1797974364635126D-03,-8.9089102067615374D-06/
      PARAMETER(NGAM=20, NCSC= 9)

C   DATA statements for target accuracies up to 1.0E-07
      DATA GAM /
     X -5.77215665E-01,-5.77215665E-01, 1.42488689E+00,-9.37711577E-01,
     X -9.77347727E-02, 6.13061586E-01, 1.03554835E-01, 3.83345368E-01,
     X  1.67609853E-01, 3.42315307E-01, 1.87209638E-01, 3.04417645E-01/
      DATA CSC, PI2 /
     X  1.66666667E-01,-1.16666667E-01, 1.12244898E-02,-2.83962070E-02,
     X  3.88183101E-03,-8.90891021E-06/
      PARAMETER(NGAM=11, NCSC= 5)
```

Fig. 5. Output of program ZETA for accuracies $10^{-14}$ and $10^{-7}$.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     Machine-dependent parameters :                                   C
C                                                                      C
C     ACC       convergence criterion for continued fractions.         C
C               Except near zero-crossings of the functions,           C
C               the relative errors of the returned functions should be C
C               less than max(ACC, (50 + imag(Z))*unit-roundoff )      C
C                                                                      C
C     FPMAX     magnitude of largest floating point number * ACC       C
C     FPMIN     magnitude of smallest floating point number / ACC      C
C     FPLMIN    ln(FPMIN)                                              C
C     FPHMIN    sqrt(FPMIN)                                            C
C     LIMIT     max. no. iterations for CF1, CF2 continued fractions   C
C               (If XNU+NL > 0.35*|Z|,  then |Z| is limited to LIMIT). C
C                                                                      C
C     GAM, CSC are the coefficients of the continued fraction form     C
C               of the diagonal Pade approximants for                  C
C               ln(Gamma(1+nu))/nu and  1/sin(pi.nu) - 1/(pi.nu) resp. C
C               The number of terms required is a linear function of   C
C                  the number of digits accuracy, i.e. of log(ACC)  .  C
C      The given GAM & CSC parameters are sufficient for ACC > 1D-24   C
C      For fixed accuracy worse than this, expressions in the code     C
C      involving ACC may be pre-evaluated, and NGAM & NCSC reduced.    C
C                                                                      C
C     Associated routine  :   CF2E (appended)                          C
C                                                                      C
C     Intrinsic functions :   MIN, MAX, SQRT, REAL, IMAG, LOG, EXP,    C
C      (Generic names)        ABS, MOD, SIN, COS, INT                  C
C              Complex    :   LOG, EXP, SIN, SQRT, DCMPLX              C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

Fig. 6. Subroutine BESSCC, machine-dependent parameters.

BESSCC, provided that sufficient rational Bernoulli numbers have been included in the DATA statements of the subroutine LOGAM. We have included 15 such numbers, sufficient for target accuracies ACC as small as $10^{-35}$; another 15 numbers are given in table 23.2 of ref. [6] if required. Fig. 5 shows the results of running the ZETA program for accuracies $10^{-7}$ and $10^{-14}$ in a FORTRAN form ready for insertion into the BESSCC deck.

## 5. Machine variants

### *5.1. The version in the CPC program package*

The published version of BESSCC is for compilation with the IBM VS FORTRAN 77 compiler, with double-precision complex type (COMPLEX * 16) available and the generic functions of fig. 6. Unfortunately, FORTRAN 77 does not define a DOUBLE COMPLEX standard, so, for convenience, statement functions for AIMAG and CMPLX have been defined.

### *5.2. Changes necessary for different machines and different precisions*

Variants decks have been prepared for the following systems and precisions:

(a) single precision for CRAY, CYBER 205 and CDC 7600, with 14 digits of precision (ACC > 1E − 14),
(b) single precision with the pure FORTRAN 77 standard, for use with IBM and GEC compliers and ACC > 1E − 7,
(c) quadruple precision for IBM VS-FORTRAN, ACC > 1Q − 24,
(d) Lahey's F77L v2.20 FORTRAN 77 for IBM PC/AT or Zenith 241/248.

The differences between these decks and the CPC version are can be summarised by:

(a, b) Remove AIMAG and CMPLX statement functions (lines 1290, 1300)
Change D+ and D− exponents to E+ and E− (this can be done by a global string substitution)
Change REAL * 8 to REAL and COMPLEX * 16 to COMPLEX

(a) Reduce PARAMETERs NGAM to 20 and NSCS to 9, deleting surplus coefficients in the DATA statements.

(b) Reduce PARAMETERs NGAM to 11 and NCSC to 6, deleting surplus coefficients in the DATA statements.

(c) Change D+ and D− exponents to Q+ and Q− (this can be done by a global string substitution)
Change REAL * 8 to REAL * 16 and COMPLEX * 16 to COMPLEX * 32
Remove the CMPLX statement functions (lines 1290, 4220)

(d) Remove AIMAG statement function (lines 1300, 4230) and AIMAG from the REAL * 8 declarations (lines 900, 4200), AIMAG is now generic.

It is also necessary to change the DATA statements for FPMAX, FPMIN, FPHMIN, and FPLMIN as shown in table 1.

## Note added in proof

In the Siemens FORTRAN 77 compiler, the REAL of a COMPLEX*16 argument gives a REAL*4 result, so should be replaced in BESSCC by DREAL.

Table 1

| Machine | min ACC | FPMAX [a] | FPMIN [b] | FPHMIN [c] | FPLMIN [d] |
|---|---|---|---|---|---|
| IBM VS sp | 9.6 E−7 | 1.E60 | 1.E−60 | 1.E−30 | −140.0 |
| dp | 2.3 D−16 | 1.D60 | 1.D−60 | 1.D−30 | −140.0 |
| qp | 3.5 Q−33 | 1.Q60 | 1.Q−60 | 1.Q−30 | −140.0 |
| CDC 7600/ 26600 | 1.0 E−14 | 1.E308 | 1.E−279 | 7.E−140 | −669.0 |
| CRAY1 | 7.5 E−15 | 1.E2450 | 1.E−2450 | 1.E−1255 | −5461.0 |
| VAX (D) | 1.4 E−17 | 1.E20 | 1.E−20 | 1.E−10 | −46.0 |
| (G) | 1.2 D−16 | 1.D290 | 1.D−290 | 1.D−145 | −667.0 |
| GEC 4190 sp | 1.2 E−7 | 1.E56 | 1.E−57 | 7.E−29 | −35.0 |
| Apollo | IEEE standard | | | | |
| Zenith 241/248 running F77L | | | | | |
| sp | 1.2 E−7 | 1.E31 | 1.E−31 | 3.E−15 | −71.0 |
| dp | 1.2 D−16 | 1.D292 | 1.D−292 | 1.D−146 | −672.0 |

[a] FPMAX = maximum floating-point number * ACC.
[b] FPMIN = minimum floating-point number / ACC.
[c] FPHMIN = sqrt(FPMIN).
[d] FPLMIN = ln(FPMIN).

## References

[1] N.M. Temme, Numer. Math. 41 (1983) 63; J. Comput. Phys. 19 (1975) 324.
[2] I.J. Thompson and A.R. Barnett, J. Comput. Phys. 64 (1986) 490.
[3] A.R. Barnett, D.H. Feng, J.W. Steed and L.J.B. Goldfarb, Comput. Phys. Commun. 8 (1974) 377.
[4] J.B. Campbell, ACM Trans. Math. Software 6 (1980) 581.
[5] I.J. Thompson and A.R. Barnett, Comput. Phys. Commun. 36 (1985) 363.
[6] M. Abramowitz, Handbook of Mathematical Functions, eds. M. Abramowitz and I.A. Stegun (Nat. Bur. Std., New York, 1964) chap. 10.
[7] J.B. Campbell, Comput. Phys. Commun. 24 (1981) 97.
[8] D.E. Amos, Trans on Num. Soft 12 (1986) 265.
[9] W. Gautschi, SIAM Rev. 9 (1966) 24.
[10] D.J. Sookne, J. Res. Natl. Bur. Std. B 77A (1973) 125.
[11] M. Goldstein and R. Thaler, Math. Tables Aids Comput. 13 (1959) 102.
[12] K.S. Kölbig, Comput. Phys. Commun. 4 (1972) 221.
[13] C.J. Noble and I.J. Thompson, Comput. Phys. Commun. 33 (1984) 413.

## TEST RUN OUTPUT

```
TEST OF THE CONTINUED FRACTION  BESSEL ROUTINES


SMALLEST ACC ALLOWED ON THIS MACHINE =  2.22E-16

Z =      0.0100   0.0000, NU(MIN) =   0.2000000  NL =  11  MODE =  -2        CAMPBELL: I,K
BESSCC  :: IFAIL =    0  ACCUR =   2.2E-16  ITS =      5     0     0, KASE = 3

NU =    0.200 :: I = 3.73712478955D-01 0.00000000000D+00,  K =  5.67109935638D+00 0.00000000000D+00
NU =    1.200 :: I = 1.55712058357D-03 0.00000000000D+00,  K =  2.67561762171D+02 0.00000000000D+00
NU =    3.200 :: I = 5.52951760527D-09 0.00000000000D+00,  K =  2.82572849307D+07 0.00000000000D+00
NU =   10.200 :: I = 5.75833581320D-31 0.00000000000D+00,  K =  8.51280398481D+28 0.00000000000D+00


Z =     12.2000  13.3000, NU(MIN) =   0.1000000  NL =  31  MODE =  -2        CAMPBELL: I,K
BESSCC  :: IFAIL =    0  ACCUR =   2.2E-16  ITS =     14    17     0, KASE = 2

NU =    0.100 :: I = 8.97067490386D-02 2.91600142087D-02,  K =  1.21799426150D-01-2.67243391440D-01
NU =    1.100 :: I = 8.69698296389D-02 3.07378697847D-02,  K =  1.18086238534D-01-2.76194408262D-01
NU =    3.100 :: I = 6.85137743833D-02 3.90017257908D-02,  K =  8.33220774949D-02-3.41784204172D-01
NU =   10.100 :: I =-9.97746863581D-03 8.63426389444D-03,  K = -2.07666335500D+00-8.32522042385D-02
NU =   30.100 :: I =-5.14314589898D-10 4.51443167494D-10,  K = -2.04395729444D+07-1.23432440464D+07


Z =     12.2000  13.3000, NU(MIN) =   0.1000000  NL =  31  MODE =   1         I,K NOT SCALED
BESSCC  :: IFAIL =    0  ACCUR =   2.2E-16  ITS =     14    17     0, KASE = 2

NU =    0.100 :: I = 1.78327284932D+04 5.79669447186D+03,  K =  6.12706606217D-07-1.34435601693D-06
                 I'= 1.73791185986D+04 6.05925271341D+03,  K' =-5.97221917913D-07 1.38184674925D-06
NU =    1.100 :: I = 1.72886586090D+04 6.11035504245D+03,  K =  5.94027580754D-07-1.38938370970D-06
                 I'= 1.68459962524D+04 6.32146072237D+03,  K' =-5.74776929678D-07 1.42827875322D-06
NU =    3.100 :: I = 1.36197950513D+04 7.75311996307D+03,  K =  4.19148011930D-07-1.71933026630D-06
                 I'= 1.32965225234D+04 7.66998024387D+03,  K' =-3.67885815542D-07 1.76488482602D-06
NU =   10.100 :: I =-1.98341252067D+03 1.71639799032D+03,  K = -1.04465628182D-05-4.18796517616D-07
                 I'=-1.68776286080D+03 1.95574468615D+03,  K' = 1.07444644340D-05-1.34394922805D-06
NU =   30.100 :: I =-1.02240160746D-04 8.97420040555D-05,  K = -1.02820364325D+02-6.20921412232D+01
                 I'=-4.43279185418D-05 2.25772742541D-04,  K' = 2.03066790405D+02-2.09909921000D+01


Z =      0.0000  19.2000, NU(MIN) =   0.7280000  NL =  11  MODE =  -1        CAMPBELL: I
BESSCC  :: IFAIL =    0  ACCUR =   2.2E-16  ITS =     41     0     0, KASE = 3

NU =    0.728 :: I =-3.13641975378D-05-6.88861621055D-05,  K =  1.18653850886D-01-2.60316218170D-01
                 I'= 1.65692906800D-01-7.54407692436D-02,  K' =-1.11824784834D-01 2.63310015119D-01
NU =    1.728 :: I = 1.65695518734D-01-7.54419584695D-02,  K =  1.01954461562D-01-2.67808973632D-01
                 I'= 6.75841206471D-03 1.48437105239D-02,  K' =-9.45510432596D-02 2.69492119711D-01
NU =    3.728 :: I = 1.57239768101D-01-7.15920149532D-02,  K =  2.27662988565D-02-2.87828013924D-01
                 I'= 2.74489712304D-02 6.02870288497D-02,  K' =-1.45616295959D-02 2.83088477613D-01
NU =   10.728 :: I = 6.55051721147D-03 1.43871045947D-02,  K = -1.52194161742D-01 2.74343890690D-01
                 I'=-1.51154841677D-01 6.88215189851D-02,  K' = 1.16204527658D-01-2.33745529556D-01
```

```
Z =     -0.0100   0.0010, NU(MIN) =   0.2000000  NL =  11  MODE =  -1        I,K NEAR -Z AXIS
BESSCC  :: IFAIL =   0  ACCUR =   2.2E-16  ITS =     5     0     0, KASE = 3

NU =   0.200 :: I = 3.069642010390-01 2.138039974000-01,  K =  4.555205662240+00-4.315573714510+00
                I'=-5.656492866430+00-4.842355369490+00,  K' = 1.289456744220+02-5.053056966840+01
NU =   1.200 :: I =-1.368088232500-03-7.629436772170-04,  K = -2.276934572670+02 1.269672656740+02
                I'= 1.534837418570-01 1.069027102150-01,  K' =-2.856576516620+04 1.238426645850+04
NU =   3.200 :: I =-5.351518079410-09-1.710566918930-09,  K = -2.596606394980+07 8.299798972720+06
                I'= 1.641340875480-06 7.115162434610-07,  K' =-8.489892801280+09 1.806976868060+09
NU =  10.200 :: I = 5.607088122570-31-2.293700470010-31,  K =  7.340896970590+28 3.002953248950+28
                I'=-5.894247284890-28 1.750151265290-28,  K' = 7.110314725700+31 3.774044604300+31


Z =     -12.2000  13.3000, NU(MIN) =   0.1000000  NL =  31  MODE =  -2        I,K RE(Z) < 0
BESSCC  :: IFAIL =   0  ACCUR =   2.2E-16  ITS =    14    17     0, KASE = 2

NU =   0.100 :: I = 9.432712817550-02-1.191156536890-05,  K = -9.160888641150-02-2.818220637520-01
NU =   1.100 :: I =-9.221174733360-02 2.358295999490-03,  K = -9.656586590780-02-2.732237778830-01
NU =   3.100 :: I =-7.721266766250-02 1.592092482690-02,  K = -1.225275352270-01-2.152423702800-01
NU =  10.100 :: I =-6.821002284920-03-1.129488030950-02,  K = -2.712534006930-02 3.134514218590-02
NU =  30.100 :: I =-3.496386313980-10-5.882799149160-10,  K = -3.953974959150-04 4.568996655000-04


Z =     -12.2000  13.3000, NU(MIN) =   0.1000000  NL =  31  MODE =   1        RE(Z)<0,NO SCALING
BESSCC  :: IFAIL =   0  ACCUR =   2.2E-16  ITS =    14    17     0, KASE = 2

NU =   0.100 :: I = 1.875120973980+04-2.367889968460+00,  K = -1.821085276690+04-5.602316882650+04
                I'=-1.840093605230+04 3.922487827510+02,  K' = 1.903570381170+04 5.459811131620+04
NU =   1.100 :: I =-1.833069497780+04 4.688036598830+02,  K = -1.919624651320+04-5.431392287750+04
                I'= 1.797493330180+04-8.063672833630+02,  K' = 1.985945456440+04 5.292325806760+04
NU =   3.100 :: I =-1.534904066210+04 3.164907131760+03,  K = -2.435714471930+04-4.278784807790+04
                I'= 1.501589863190+04-3.185733264960+03,  K' = 2.409595358640+04 4.177225747640+04
NU =  10.100 :: I =-1.355941254160+03-2.245299668990+03,  K = -5.392223326830+03 6.231074207600+03
                I'= 1.000799522040+03 2.381571134430+03,  K' = 6.144153127660+03-5.302263402440+03
NU =  30.100 :: I =-6.950436674250-05-1.169436649210-04,  K = -7.860073257690+01 9.082669666230+01
                I'=-2.760925852880-05 2.284207181500-04,  K' =-1.996138582600+02 4.278733014790+01


Z =      0.0000 500.2000, NU(MIN) =   0.7280000  NL =   9  MODE =  -2        LARGE IM(Z)
BESSCC  :: IFAIL =   0  ACCUR =   2.2E-16  ITS =     0     6     3, KASE = 1

NU =   0.728 :: I =-4.775029778490-03-1.048754634910-02,  K = -5.502539333880-03 5.576791494160-02
NU =   1.728 :: I =-3.070216321750-02 1.397884106990-02,  K = -5.365624958130-03 5.578139257770-02
NU =   3.728 :: I =-3.058545504880-02 1.392570328520-02,  K = -4.756924416800-03 5.583720840140-02


Z =      0.0000   0.0010, NU(MIN) =   0.0000001  NL =   9  MODE =  -2        A COULCC ERROR
BESSCC  :: IFAIL =   0  ACCUR =   2.2E-16  ITS =     4     0     0, KASE = 3

NU =   0.000 :: I = 9.999990476320-01 1.570794830820-07,  K =  7.023684788720+00-1.570795934100+00
NU =   1.000 :: I =-7.853974350440-11 4.999995363160-04,  K = -9.424777693180-04-1.000004464210+03
NU =   3.000 :: I = 3.272489244510-18-2.083331357910-11,  K =  1.256638292490+03 8.000007818950+09
```