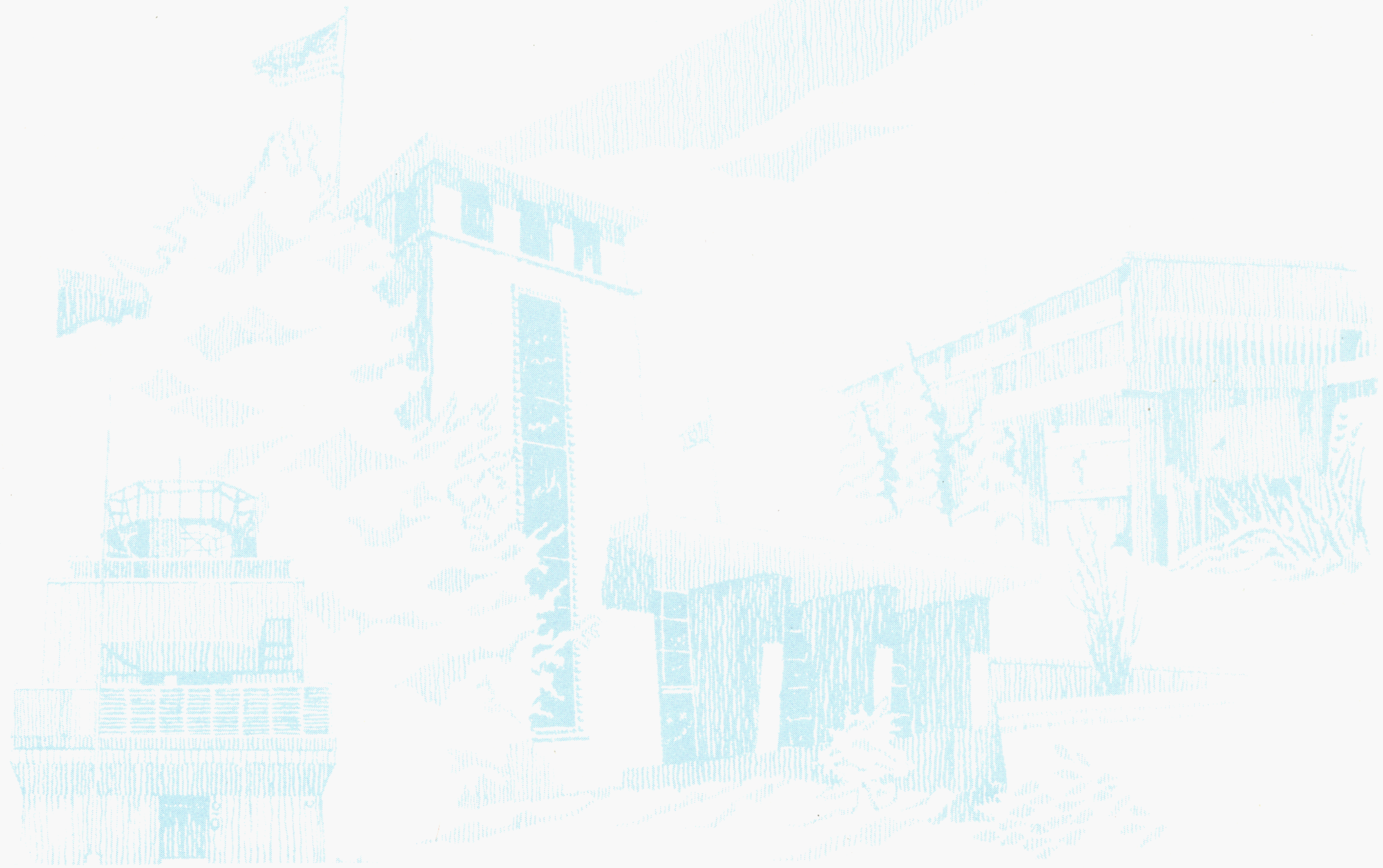
SANDIA REPORT SAND85—1018 • Unlimited Release • UC—32

Printed May 1985

A Subroutine Package for Bessel Functions of a Complex Argument and Nonnegative Order

Donald E. Amos

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789



Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A02
Microfiche copy: A01

UC-32
SAND85-1018
UNLIMITED RELEASE
PRINTED MAY 1985

A SUBROUTINE PACKAGE FOR BESSEL FUNCTIONS OF A COMPLEX ARGUMENT AND NONNEGATIVE ORDER

by

Donald E. Amos
Numerical Mathematics Division 1642

Abstract

The machine implementation of a Bessel function package, which was outlined in mathematical form in two previous documents, is described for Bessel functions $H_\nu^{(1)}(z)$, $H_\nu^{(2)}(z)$, $I_\nu(z)$, $J_\nu(z)$, $K_\nu(z)$, $Y_\nu(z)$ and Airy Functions $Ai(z)$, $Ai'(z)$, $Bi(z)$, $Bi'(z)$ with $\nu \geqq 0$ and $-\pi < \arg z \leqq \pi$. The less obvious aspects of the implementation are the main topics of interest. These topics include estimation of function magnitudes for underflow and overflow tests, scaling near underflow and overflow limits, the use of recurrence in connection with scaling, the use of machine constant routines and an error package for portability.

1. Introduction

In [2] and [3], the analytic basis for the construction of a Bessel function package for non-negative orders and complex arguments was outlined. The implementation of these schemes in a portable fashion poses some machine oriented problems. This document explains the less obvious problems and their solutions. These include (1) estimation of function magnitudes for the determination of underflow and overflow, (2) scaling near underflow and overflow limits, (3) the use of recurrence in connection with scaling, (4) the influence of small exponent ranges on the package, (5) the use of machine constant routines, and (6) the use of an error package for printing error messages.

More precisely, the outline in [2] and [3] sets forth the mathematical basis for evaluating Bessel functions $H_\nu^{(1)}(z)$, $H_\nu^{(2)}(z)$, $I_\nu(z)$, $J_\nu(z)$, $K_\nu(z)$, $Y_\nu(z)$ and Airy functions $Ai(z)$, $Ai'(z)$, $Bi(z)$, $Bi'(z)$ for $\nu \geqq 0$ and $-\pi < \arg z \leqq \pi$. The basic idea is to compute $I_\nu(z)$ and $K_\nu(z)$ in the right half plane and relate all other functions, including their analytic continuations, to these two functions. Figures 1 and 2 show the regions where different formulae are applied. The callable routines in the package are listed below.

Function	Single Precision Name	Double Precision Name	Quick Check Drivers
$H_\nu^{(1)}(z)$, $H_\nu^{(2)}(z)$	CBESH	ZBESH	CQCBH, ZQCBH
$I_\nu(z)$	CBESI	ZBESI	CQCBI, ZQCBI
$J_\nu(z)$	CBESJ	ZBESJ	CQCBJ, ZQCBJ
$K_\nu(z)$	CBESK	ZBESK	CQCBK, ZQCBK
$Y_\nu(z)$	CBESY	ZBESY	CQCBY, ZQCBY
$Ai(z)$, $Ai'(z)$	CAIRY	ZAIRY	CQCAI, ZQCAI
$Bi(z)$, $Bi'(z)$	CBIRY	ZBIRY	
$\ln \Gamma(x)$, $x > 0$	GAMLN	DGAMLN	

The quick check drivers are executable programs which evaluate a variety of relations. These results are checked against known values or values computed from the package. These drivers are provided to give first-time users confidence that the package has been installed properly. All routines, except the Airy function routines and the quick check drivers, can return N member sequences for orders $\nu, \nu+1, \ldots, \nu+N-1$. The package also allows for exponential scaling to increase the argument range of each function. A list of lower level routines and their purpose is shown in Section 8.

Since there is no standard type DOUBLE PRECISION COMPLEX, the double precision routines carry complex numbers as ordered pairs. Section 8 also lists the common mathematical functions needed to manipulate double precision ordered pairs in calling programs.

The package was written according to FORTRAN 66 standards and passed the PFORT analyzer. Except possibly for hollerith types, which occur only in error messages from calls to SUBROUTINE XERROR, the package should compile on many FORTRAN 77 compilers.

The sections to follow detail some of the more important considerations for implementation of basic formulae.

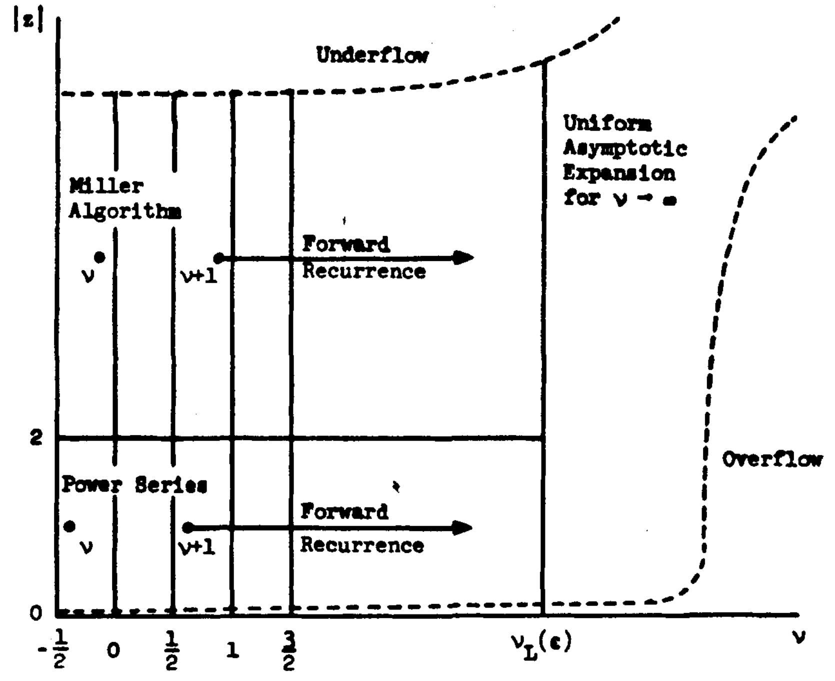


Figure 1. Computation Diagram for $K_\nu(z)$, $\mathrm{Re}(z) \geqq 0$, $z \neq 0$

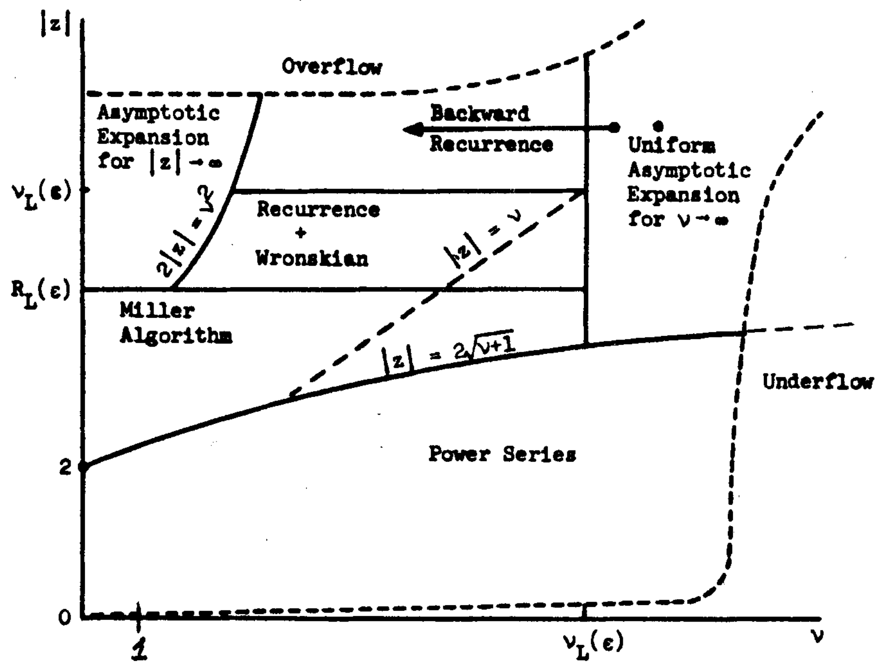


Figure 2. Computation Diagram for $I_\nu(z)$, $\mathrm{Re}(z) \geqq 0$

2. Estimation of Magnitudes of $I_\nu(z)$ and $K_\nu(z)$, $\mathrm{Re}(z) \geqq 0$

The magnitudes of the I and K functions are needed for overflow and underflow tests. These tests are made in logarithmic form so that the computations themselves will remain on scale. That is,

$$|\ln I_\nu(z)| \leqq \mathrm{ELIM} \qquad \text{or} \qquad |\ln K_\nu(z)| \leqq \mathrm{ELIM}$$

must be satisfied in order to proceed, where ELIM is the package exponential overflow or underflow limit,

$$e^{\pm \mathrm{ELIM}} = \begin{cases} \text{largest (positive) package number} \\ \text{smallest (positive) package number.} \end{cases}$$

ELIM is computed in terms of machine constants and is described in Section 6.

It was determined experimentally that the correct order of magnitude for both the I and K functions could be obtained from the leading terms of the uniform expansions for $\nu \to \infty$ for orders $\nu \geqq 1$ with surprising accuracy, even for orders as low as $\nu = 1$. Recall [3, pp.

22-23] that different formulae are applied in the ranges $0 \leq \arg z \leq \frac{\pi}{3}$, $\frac{\pi}{3} < |\arg z| \leq \frac{\pi}{2}$:

$$\left.\begin{aligned} I_\nu(z) &\sim \sqrt{\frac{t}{2\pi\nu}}\, e^{\nu\xi}\,(1 + \Sigma_1) \\ K_\nu(z) &\sim \sqrt{\frac{\pi t}{2\nu}}\, e^{-\nu\xi}\,(1 + \Sigma_2) \end{aligned}\right\} \qquad 0 \leq |\arg z| \leq \frac{\pi}{3} \qquad (2.1)$$

$$\xi = -\left(\ln\left[\frac{1 + \sqrt{1 + (z/\nu)^2}}{z/\nu}\right] - \sqrt{1 + (\tfrac{z}{\nu})^2}\right), \quad t = \frac{1}{\sqrt{1 + (\frac{z}{\nu})^2}}$$

and

$$\left.\begin{aligned} I_\nu(z) &= e^{\mp i\pi\nu/2}\, J_\nu(ze^{\pm i\pi/2}) \\ K_\nu(z) &= \pm\frac{i\pi}{2}\, e^{\pm i\pi\nu/2} \begin{cases} H_\nu^{(1)}(ze^{i\pi/2}) \\ H_\nu^{(2)}(ze^{-i\pi/2}) \end{cases} \end{aligned}\right\} \qquad \frac{\pi}{3} < |\arg z| \leq \frac{\pi}{2}$$

where

$$\begin{aligned} J_\nu(z) &\sim \phi(z)\left\{\mathrm{Ai}(\alpha)S_1 + \frac{\mathrm{Ai}'(\alpha)}{\nu^{4/3}}\, S_2\right\} \\ H_\nu^{(1)}(z) &\sim 2e^{-\pi i/3}\phi(z)\left\{\mathrm{Ai}(\beta_1)S_1 + \frac{e^{2\pi i/3}\mathrm{Ai}'(\beta_1)}{\nu^{4/3}}\, S_2\right\} \qquad (2.2) \\ H_\nu^{(2)}(z) &\sim 2e^{\pi i/3}\phi(z)\left\{\mathrm{Ai}(\beta_2)S_1 + \frac{e^{-2\pi i/3}\mathrm{Ai}'(\beta_2)}{\nu^{4/3}}\, S_2\right\} \end{aligned}$$

$$\alpha = \nu^{2/3}\zeta, \quad \beta_1 = \alpha e^{2\pi i/3}, \quad \beta_2 = \alpha e^{-2\pi i/3}, \quad \phi(z) = \left[\frac{4\zeta}{1 - (z/\nu)^2}\right]^{1/4} \Big/ \nu^{1/3}$$

$$\frac{2}{3}\zeta^{3/2} = \ln\left(\frac{1 + \sqrt{1 - (z/\nu)^2}}{z/\nu}\right) - \sqrt{1 - (z/\nu)^2} .$$

Here $\sum_1$, $\sum_2$, S_1 and S_2 are sums in reciprocal powers of ν. At this point, we are interested in only the leading terms and we simplify to obtain

$$I_\nu(z) \sim \sqrt{\frac{t}{2\pi\nu}}\; e^{\nu\xi} \quad , \qquad K_\nu(z) \sim \sqrt{\frac{\pi t}{2\nu}}\; e^{-\nu\xi}$$

$$J_\nu(z) \sim \frac{\phi(z)}{2\sqrt{\pi}}\, \frac{e^{-(2/3)\alpha^{3/2}}}{\alpha^{1/4}} \, , \qquad H_\nu^{(1)}(z) \sim \frac{2e^{-i\pi/3}\phi(z)e^{-(2/3)\beta_1^{3/2}}}{2\sqrt{\pi}\quad \beta_1^{1/4}} \tag{2.3}$$

$$H^{(2)}(z) \sim \frac{2e^{+i\pi/3}\phi(z)e^{-(2/3)\beta_2^{3/2}}}{2\sqrt{\pi}\quad \beta_2^{1/4}}$$

using

$$Ai(z) \sim \frac{1}{2\sqrt{\pi}}\, \frac{e^{-(2/3)z^{3/2}}}{z^{1/4}}$$

The component parts of (2.1) and (2.2) are programmed in subroutines CUNIK and CUNHJ and the magnitudes according to (2.3) are programmed in CUOIK. The parameter IKFLG chooses the I or K function in CUNIK while IPMTR in the call to CUNIK or CUNHJ selects the quantities needed for (2.3) or the quantities needed for (2.1) and (2.2).

In routines for small $|z|$ where power series are used, different tests are made. For the I function in CSERI where $(|z|/2)^2 \leq \nu + 1$, the exponential in

$$I_\nu(z) \sim \frac{(z/2)^\nu}{\Gamma(1+\nu)} = e^{\nu \ln(z/2) - \ln \Gamma(\nu+1)}$$

is tested for underflow. For $K_\nu(z)$, $-1/2 \leq \nu < 1/2$, the exponential in

$$K_\nu(z) \sim \frac{\pi}{2 \sin \pi\nu} \left[\frac{(z/2)^{-\nu}}{\Gamma(1-\nu)} - \frac{(z/2)^{\nu}}{\Gamma(1+\nu)} \right]$$

remains on scale. However, when sequences of K functions are needed, the recurrence is started with K_ν and $K_{\nu+1}$, $-1/2 \leqq \nu < 1/2$, and

$$K_{\nu+1}(z) \sim \frac{1}{2} (\frac{2}{z})^{\nu+1} \Gamma(1 + \nu).$$

Thus, $K_{\nu+1}$ can overflow when $|z|$ is small since $\nu + 1$ can be as much as 3/2. This test is made on $(\nu + 1)|\ln(2/z)|$ in CBESH and CBESK when appropriate. In ranges where $|z|$ is large ($|z| > \max(R_L(\varepsilon),\frac{\nu^2}{2})$), the expansion for $z \to \infty$ is applied for $I_\nu(z)$. For $K_\nu(z)$ in $|z| > 2$ and $-\frac{1}{2} \leqq \nu < \frac{1}{2}$, the leading terms of the Miller algorithm is asymptotically correct (Figures 1 and 2) and one has

$$I_\nu(z) \sim \frac{e^z}{\sqrt{2\pi z}} \quad , \quad K_\nu(z) \sim \sqrt{\frac{\pi}{2z}}\, e^{-z} \quad , \quad |z| \to \infty$$

or

$$|\ln I_\nu(z)| \sim x - \frac{1}{2} \ln(2\pi|z|) \quad , \quad |\ln K_\nu(z)| \sim - x - \frac{1}{2} \ln(2|z|/\pi)$$

When N member sequences are involved, underflow tests are made on the last member of the I sequence and the first member of the K sequence; while the reverse is true for overflow tests. In the event that the last member of a K sequence underflows, then the whole sequence has underflowed and NZ is set to N. However if the first member has underflowed and the last member is on scale, then recurrence is carried forward scaled by e^x until two members come on scale. Then recurrence is continued in a normal fashion. (Here "normal fashion" means the procedure described in Section 3.) NZ is set to the number of the underflowed members whose values are set to zero before leaving the subroutine. Scaled recurrence is carried out in subroutine CKSCL. Similarly for I sequences. Recurrence is carried in a backward fashion from the last member. If a member underflows, it is set to zero and succeeding members are tried until two members come on scale. Then backward recurrence is used to finish the sequence (as described in Section 3.) NZ is set to the number of zero entries in the sequence.

3. Scaling Near Underflow and Overflow Limits

The $I_\nu(z)$ function underflows when ν is large compared with $|z|$. This may occur in subroutines CSERI, CUNI1, and CUNI2 where the power series and the two uniform expansions (2.1) and (2.2) for $\nu \to \infty$ are used. The $K_\nu(z)$ function underflows (exponentially) when $|z|$ is large compared to ν. This may occur in CBKNU, CUNK1 or CUNK2. The problem with computation near underflow and overflow limits can be illustrated by examples. An expansion typically has the form $c(1 + \varepsilon)$ where c displays the dominant behavior. Consider

$$(10^{-290} + 10^{-290}i)(1 + 10^{-5}i) = (10^{-290} - 10^{-295}) + i(10^{-290} + 10^{-295})$$

on a CDC machine where underflow is 10^{-294} and unit round off is 10^{-14}. Here 10^{-295} is an underflow and the result is in error in the 5th digit. Notice, however, that scaling by the reciprocal of unit round off (10^{14}) can keep the multiplication on scale and yield correct results:

$$10^{-14}(10^{-276} + 10^{-276}i)(1 + 10^{-5}i)$$

$$= 10^{-14}[(10^{-276} - 10^{-281}) + i(10^{-276} + 10^{-281})]$$

$$= 10^{-14}(.99999 \times 10^{-276} + 1.00001 \times 10^{-276}i)$$

$$= .99999 \times 10^{-290} + 1.00001 \times 10^{-290}i$$

Thus, 10^{-5} could be replaced by anything smaller and still obtain answers accurate to within unit round off.

This example shows the case where the real and imaginary parts remain on scale. However, consider the final scaling operation

$$10^{-14}(10^{-276} + 10^{-281}i) = 10^{-290} + 10^{-295}i.$$

In this example the imaginary part underflows leaving both the magnitude and phase in error in the 5th digits. The only reasonable recourse is to declare this an underflow and return a value of zero with an error flag indicator. Thus, in order to be successful at scaling with a potential underflow in one of the components we must have the larger component at least one precision = 1/TOL larger than the smaller component. This will give a magnitude with relative accuracy and a phase angle with absolute accuracy. These requirements are consistent with what is obtainable with complex arithmetic in general. That is, relative accuracy in the phase angle is not always possible when one component dominates because of subtractions in a complex multiply. These underflow tests on components are made in CUCHK before the final scaling by TOL is done. CUCHK returns a non-zero error flag on an underflow which is processed by the calling routine to return a zero value for the function.

For overflow, these problems are mitigated somewhat because of the inequalities

$$z = x + iy$$

$$|z| \geqq |x| \quad , \quad |z| \geqq |y| ,$$

That is, a magnitude which does not overflow ensures that the real and imaginary parts do not overflow. This does not mean that a result which is on scale can be computed; intermediate results could overflow.

To cope with three kinds of problems, a scaling factor which can take on three different values is introduced. The first, 1/TOL = reciprocal of unit round off, is applied when the magnitude of the I or K function is within 1/TOL of the underflow limit. The second value, equal to 1, is applied in the intermediate range of values, while the third value, TOL = unit round off, is applied when the magnitude is within TOL of the overflow limit. Scaled values are carried during recurrence with multiplication by the reciprocal of the scale factor being the last step (we know from Section 2 that the magnitude will be on scale and subroutine CUCHK guarantees that the phase can be computed accurately near underflow limits.)

4. Analytic Continuation of $K_\nu(z)$, $\mathrm{Re}(z) > 0$

The analytic continuation of $K_\nu(z)$ from the right to the left half plane is carried out by the formula

$$K_\nu(ze^{\pm i\pi}) = e^{\mp i\pi\nu} K_\nu(z) \mp i\pi I_\nu(z) \quad , \qquad \mathrm{Re}\ (z) \geq 0 \qquad (4.1)$$

in subroutine CACON except when ν > FNUL where the uniform expansions is used. The package allows exponential scaling for most of the functions and (4.1) applies for KODE = 1. The scaled version for KODE = 2 is

$$e^{-z}K_\nu(ze^{\pm i\pi}) = e^{\mp i\pi\nu} e^{-2z}\{e^{z}K_\nu(z)\} \mp i\pi e^{-iy}\{e^{-x}I_\nu(z)\} \qquad (4.2)$$

In (4.1) the I and K functions are of different orders of magnitude and no problem is encountered with addition. In particular, underflow cannot occur. On the other hand, the terms on the right of (4.2) can be of similar magnitude and are generally smaller than those in the unscaled version (4.1). Provision is made for one or both terms to underflow. To avoid the problems outlined in Section 3, each term must be larger (in magnitude) than the underflow limit scaled by 1/TOL in order to be added together. This test is done in CS1S2.

One can see from (4.1) that neither forward nor backward recurrence is appropriate for the left half plane. For large z, $I_\nu(z)$ dominates and one would use backward recurrence for stability. On the other hand, for large ν, K_ν dominates and one would use forward recurrence for stability. Consequently, the sequences for the right side of (4.1) are generated for $\mathrm{Re}(z) \geq 0$ and (4.1) or (4.2) is applied to each member of the sequence.

5. Avoiding Recursive Calls

The analytic continuation of $I_\nu(z)$ from the right to the left half z plane is carried out by the formula

$$I_\nu(ze^{\pm i\pi}) = e^{\pm i\pi\nu} I_\nu(z) , \qquad \mathrm{Re}\,(z) \geqq 0 \qquad (5.1)$$

in subroutine CBESI. However $I_\nu(z)$ for $\mathrm{Re}(z) \geqq 0$ is computed in CBINU. Now, CBINU calls routines CBUNI, CUNI1, and CUNI2 which in turn call CAIRY for the Airy functions Ai and Ai' when the order is large. Now Ai and Ai' are computed from $K_{1/3}$ and $K_{2/3}$ in CBKNU when arguments are in the right half plane and possbily by (4.1) or (4.2) when the arguments are in the left half plane. Recall that CACON evaluates (4.1) or (4.2) which needs $I_\nu(z)$ for $\mathrm{Re}(z) \geqq 0$. Thus, the construction establishes a path which expresses I_ν in the right half plane in terms of I_ν in the right half plane, a situation which would naturally result in a recursive call to CBINU.

Another potential for a recursive call exists in the analytic continuation of the K function by (4.1) in CACON. For large orders, CACON calls CBINU which, farther down the line, can call CUNI2. Now, CUNI2 computes by (2.2) which calls the Airy Functions Ai and Ai' from CAIRY. These in turn require K functions of orders 1/3 and 2/3 and their analytic continuations, which would naturally be computed by calls to CACON. Hence a recursive call is possible.

Notice that the problem in both situations occurs because of calls to the Airy routine. Both problems are solved if the Airy routine has its own continuation subroutine. Thus, we construct a continuation subroutine CACAI from CACON by retaining only those parts which apply to orders $\nu \leqq 1$ and deleting all other parts (see Figures 1 and 2 for the formulae when $\nu \leqq 1$.) This leads CACAI to call CSERI, CMLRI, and CASYI for $I_{1/3}$ and $I_{2/3}$ in the right half z plane so that (4.1) and (4.2) can be applied.

6. Machine Related Parameters and the Error Package

Included in the package are function subroutines which define the machine environment in which the package is to operate [5]. This helps make the package portable by relegating all machine dependent parameters to these function subroutines. These routines [5] are called I1MACH, R1MACH and D1MACH.

I1MACH defines integer constants associated with the environment. For example, standard input, output, punch and error message units are defined by I1MACH(I), I = 1, 4. The number of bits per storage unit (word) and number of alpha-numeric characters in a storage unit are defined in I1MACH(5) and I1MACH(6). The constants associated with integer, single precision and double precision arithmetic are defined in I1MACH(7) through I1MACH(16). These include the base of the arithmetic B, maximum and minimum exponents and the number of base B digits.

R1MACH(I), I = 1, 5 returns the smallest and largest (positive) floating (single precision) numbers, the smallest relative spacing, the largest relative spacing (unit round off) and log base 10 of the single precision arithmetic base. Similarly for D1MACH(I), I = 1,5 for double precision arithmetic.

To make the usage easier, these quantities are defined on comment cards for a wide variety of systems. To define one of these systems, one has only to replace C's in column 1 by spaces to make a particular set of FORTRAN statements active.

In the main routines, which are called by the user, package parameters TOL, ELIM, and ALIM are computed from machine constants and passed to lower level routines. These package parameters are slightly different from those which could be computed directly from I1MACH, R1MACH and D1MACH because we wish to impose further limitations. Thus,

```
TOL = AMAX1(R1MACH(4), 1.0E-18)
```

defines the package unit round off limited to 18 digits because constants are stored to only 18 digits (UNIVAC double precision.) The statements

```
K = MIN0(IABS(I1MACH(12)), IABS(I1MACH(13))
ELIM = 2.303E0 * (FLOAT(K) * R1MACH(5) - 3.0E0)
AA = 2.303E0 * R1MACH(5) * FLOAT(I1MACH(11)-1)
ALIM = ELIM + AMAX1(-AA, -41.45E0)
```

define the package exponential overflow or underflow limit ELIM cushioned by 10^3 to allow for some impreciseness in tests using first term approximations. Also, one is not always sure that the ALOG (or

DLOG) function would perform correctly on the largest or smallest machine numbers. Thus,

$$\text{ELIM} = \text{AMIN1(-ALOG(R1MACH(1)), ALOG(R1MACH(2)))}$$

may be desirable, but not prudent. ALIM is a near-underflow or near-overflow quantity which triggers a non-unit scaling option described in Section 3. More precisely, these computations express the relations

$$e^{\text{ELIM}} = B^{K} \cdot 10^{-3} \quad , \quad e^{\text{ALIM}} = e^{\text{ELIM}} * \text{TOL}$$

$$e^{-\text{ELIM}} = B^{-K} \cdot 10^{3} \quad , \quad e^{-\text{ALIM}} = e^{-\text{ELIM}}/\text{TOL}$$

where I1MACH(12) and I1MACH(13) are the minimum and maximum exponents possible for a floating number, i.e. R1MACH(1) = $B^{\text{I1MACH(12)}-1}$, where B = I1MACH(10) = floating point base, etc. Notice that the computation of ELIM is accomplished without complicated function evaluations. For double precision arithmetic, indicies 11, 12, and 13 are replaced by 14, 15 and 16 respectively, and R1MACH() is replaced by D1MACH().

ELIM and ALIM are carried this way because all tests for overflow or underflow are made on the magnitude of the logarithm of a Bessel function. ±ALIM are exponent boundaries above or below which scaling is applied as described in Section 3.

Portability is also enhanced by the use of work arrays in call lists. Since a complete restoration of all variables to their former values is not always assured in successive calls, work arrays will pass those to be saved to the calling routine and restore them on the next call. This is used in subroutine CUNIK where the coefficients of (2.1) in $\sum_1$ and $\sum_2$ are computed for either I or K and saved (the terms differ only in sign) for a subsequent call to compute the analytic continuation by formula (4.1).

TABLE I

	IFLAG value (defined by calls to XSETF() or by librarian as default)		
LEVEL	0	±1	±2
2 = FATAL	PA	P(T) A	P(T)A
1 = RECOVERABLE	R	P(T)R	P(T)A
0 = WARNING	R	P(T)R	P(T)R
-1 = ONCE	R	P-once(T)R	P-once(T)R

Note: P = print error message; P-once = print first occurrence of error message; R = RETURN from XERROR(); A = abort; (T) = trace-back if IFLAG is positive and subroutine FDUMP() is coded for a trace-back.

The SLATEC error package [7] is used to print messages. This package is an enhanced version of that in [6] which has many capabilities for printing messages, printing variables, turning off fatal diagnostics, etc. The typical call to process messages without numeric output is

```
CALL XERROR(7HMESSAGE, 7, NER, LEVEL)
```

where the message and the number of characters in the message are first. NER is an error number assigned by the author. LEVEL is used to indicate the seriousness of the message. In calls to XERROR(), we use NER = 2 and LEVEL = 1 since the message gives the pertinent information and is normally fatal (A), though the user can reset IFLAG and do something else.

The variable IFLAG in a call to subroutine XSETF(IFLAG) decides how the message will be printed when the default option (IFLAG = 2) is not used. The current value can be obtained by a call to subroutine XGETF(IFLAG). Table I [7] gives the essential information.

The sign of IFLAG determines the form of the print, if it occurs. A positive sign leads to calls to FDUMP(), which in this version simply returns, but can be programmed to give trace-back information. The abort routine XERABT() simply stops the execution with a STOP statement. The default, IFLAG = 2, will print, abort, and give an error message summary. (The default parameter IPARAM(2) = 2 is contained in a DATA statement in function J4SAVE().)

Similar usage applies to subroutines XGETUN(LUN) and XSETUN(LUN) where LUN is the unit number on which error messages are to appear. The default is LUN = I1MACH(4), the standard error message unit defined by the machine integer function I1MACH() described above. XGETUA(IUNIT, N) and XSETUA(IUNIT,N) serve the same purpose when error messages are to appear on multiple units IUNIT(1),..., IUNIT(N), $N \geq 5$.

In an institutional setting the package can be tailored to the system by the librarian to provide trace back information or appropriate diagnostic information for the user from FDUMP() or XERABT().

7. Performance With High Accuracy on Low Exponent Machines

The main consideration in all routines is to achieve the accuracy specified by TOL (unit round off.) This is not always possible because of argument reduction in elementary functions when variables are large. However, we can program to ensure that unnecessary losses do not occur and maximum use is made of the full exponent range of the machine.

Asymptotic expansions require a variable to be large. Specifically, ν must be large to apply the uniform expansions (2.1) and (2.2). This makes $I_\nu(z)$ small when $|z|$ is only moderately smaller than ν. In the case of the VAX, where unit round off is approximately 10^{-16} in double precision arithmetic and the arithmetic range is only $10^{\pm 38}$, the lower bound on ν for (2.1) and (2.2), called FNUL, is approximately 88. In this case, $I_\nu(z)$ can underflow easily for $\nu >$ FNUL. This happens in CBUNI on a call from CBINU. CBINU sets all members of a sequence to zero whose orders exceed FNUL and returns the last index, NLAST, whose value was not set. At this point CUOIK gets a chance to examine the corresponding order, ν_0 + NLAST-1, to see whether or not $I_{\nu_0+\text{NLAST}-1}(z)$ is on scale. If not, then CUOIK sets more values to zero until either the sequence is exhausted or a member comes on scale. If a member is deemed to be on scale by CUOIK, then other routines get to compute it. In the case $\nu \leq$ FNUL, CUOIK also gets a chance to set underflow values before any computation is attempted. The point here is that the small exponent range forces the underflow boundary for $I_\nu(z)$ (overflow boundary for $K_\nu(z)$) to be superimposed over formula boundaries to exclude the use of the uniform expansions in some range of z. A similar situation can occur if a sequence for $I_{\nu+k}(z)$, k + N-1,..., 0 crosses the boundary $(|z|/2)^2 \leqq \nu + 1$. The last NZ members are set to zero in CSERI, the index is modified to N - NZ and the new (sub) sequence is examined for underflow by calls to CUOIK or CBUNI.

This illustrates that underflow on a VAX may be very likely because the orders ν do not have to be very large for this to happen. It is especially important that the scaling described in Section 3 be used to keep as much of the exponent range as possible; otherwise, the underflow and overflow limits (in double precision arithmetic) would have to be modified by $10^{\pm 16}$, making the effective exponent range (where accurate results can be obtained) only $10^{\pm 22}$.

8. Subroutine Names and Purpose

The main callable routines of the package are listed in Section 1. This section lists lower level routines, their purpose and the possibility of a call if one wishes to by-pass argument checking and overflow or underflow checking. Unless one has complete information on the range of variables, calls to lower level routines are not recommended.

The names of double precision versions are preceeded by a Z in place of a C. Since a type DOUBLE PRECISION COMPLEX is not available as a standard FORTRAN type, complex numbers are carried as ordered pairs. This necessitates FORTRAN subroutines or functions for a variety of the common mathematical functions. These are listed at the bottom of Table II.

TABLE II

SUBROUTINE	COMPUTATION	CALLABLE
GAMLN	$\ln \Gamma(x)$, $x > 0$	YES
CBINU	$I_\nu(z)$, $\mathrm{Re}(z) \geqq 0$	YES
CBKNU	$K_\nu(z)$, $\mathrm{Re}(z) \geqq 0$	YES
CRATI	$I_{\nu+1}(z)/I_\nu(z)$, $\mathrm{Re}(z) \geqq 0$ for CMLRI, CWRSK	YES
CSHCH	sinh z, cosh z for CBKNU	YES
CSERI	$I_\nu(z)$ by power series $(\|z\|/2)^2 \leqq \nu+1$, $\mathrm{Re}(z) \geqq 0$	NO
CMLRI	$I_\nu(z)$ by Miller algorithm normalized by a Series, $\mathrm{Re}(z) \geqq 0$	NO
CASYI	$I_\nu(z)$ by asymptotic expansion for $z \to \infty$, $\mathrm{Re}(z) \geqq 0$	NO
CWRSK	$I_\nu(z)$ by Miller algorithm normalized by Wronskian, $\mathrm{Re}(z) \geqq 0$	NO
CBUNI	Driver for CUNI1 and CUNI2	NO
CUNI1	$I_\nu(z)$ by (2.1)	NO
CUNI2	$I_\nu(z)$ by (2.2)	NO
CBUNK	Driver for CUNK1 and CUNK2	NO
CUNK1	$K_\nu(z)$ by (2.1), (4.1), (4.2)	NO
CUNK2	$K_\nu(z)$ by (2.2), (4.1), (4.2)	NO
CACON	Analytic continuation of $K_\nu(z)$ to left half plane	NO
CACAI	Analytic continuation of Ai, Ai' to left half plane	NO
CUNIK	Parameters for (2.1) for use in CUNI1, CUNK1, CUOIK	NO
CUNHJ	Parameters for (2.2) for use in CUNI2 CUNK2, CUOIK	NO
CUOIK	Set underflow or overflow indictators and set underflow values for I sequences	NO

CUCHK	Checks for component underflow when magnitude $\leqq \exp(-\mathrm{ALIM}) = 1.0\mathrm{E}{+3} * \mathrm{R1MACH(1)/TOL}$	NO
CKSCL	Scaled recurrence on underflow for CBKNU, CUNK1, CUNK2	NO
CS1S2	Addition of I and K functions for (4.2) on KODE = 2	NO

Additional Double Precision Routines

ZMLT	$z_1 * z_2 = z_3$	YES
ZDIV	$z_1/z_2 = z_3$	YES
ZSQRT	$\sqrt{z}, \qquad -\pi < \arg z \leqq \pi$	YES
ZEXP	e^z	YES
ZLOG	$\ln z, \qquad -\pi < \arg z \leqq \pi$	YES
ZSHCH	$\sinh z, \cosh z$	YES

9. Package Accuracy

The approximate relative error in the magnitude of a complex Bessel function can be expressed by P*1.0E + S where P = max(unit round off, 1.0E - 18) is the nominal precision and 1.0E + S represents the increase in error due to argument reduction in the elementary functions. Here, S = max(1, |LOG10(|z|)| , |LOG10(ν)|) approximately (i.e. S = max(1, |exponent of |z||, |exponent of ν|). However, the phase angle may have only absolute accuracy. This is most likely to occur when one component (in absolute value) is larger than the other by several orders of magnitude. If one component is 1.0E + K larger than the other, then one can expect only max(|LOG10(P)| -K, 0) significant digits; or, stated another way, when K exceeds the exponent of P, no significant digits remain in the smaller component. However, the phase angle retains absolute accuracy because, in complex arithmetic with precision P, the smaller component will not (as a rule) decrease below P times the magnitude of the larger component. In these extreme cases, the principal phase angle is on the order of $\pm$ P or $\pm \pi/2 \mp$ P.

10. Quick Check Routines

The SLATEC library standards [5] require that driver routines, which exercise major loops in each subroutine, be written. These are generally not definitive tests, but give the first-time user of the package some confidence that the package is operating correctly on his machine. A check of the Wronskian is made in many of these routines. Where recurrence is used, sequences which cross formula boundaries are checked by different formulae. These checks are made in driver routines CQCAI, CQCBH, CQCBI, CQCBJ, CQCBK, CQCBY and their corresponding double precision versions.

A technical detail arose during the evaluation of the Wronskian

$$Ai(ze^{2\pi i/3})\ Ai'(z) - e^{2\pi i/3}\ Ai'(ze^{2i\pi/3})\ Ai(z) = \frac{-e^{-\pi i/6}}{2\pi}$$

using the scaled version (KODE = 2)

$$e^{-\zeta_1-\zeta_2}[e^{\zeta_1}Ai(ze^{2\pi i/3})]\ [e^{\zeta_2}Ai'(z)]-$$

$$e^{-\zeta_1-\zeta_2}\ e^{-2\pi i/3}\ [e^{\zeta_1}Ai'(ze^{2\pi i/3})]\ [e^{\zeta_2}Ai'(z)] = \frac{-e^{-\pi i/6}}{2\pi}$$

because the scale factors are not analytic in the whole z plane and introduce a discontinuity along the line $\theta = \pi/3$:

Case I, $-\pi < \theta \leqq \pi/3$.

$$\zeta_2 = \frac{2}{3}\ z^{3/2} = \frac{2}{3}\ re^{i\theta}\ \cdot \sqrt{r}\ e^{i\theta/2} = \frac{2}{3}\ r^{3/2}\ e^{3\theta i/2}$$

since θ is in the proper range for the principal square root. Now,

$$w = re^{i\theta}\ \cdot\ e^{2\pi i/3} = re^{i(\theta+2\pi/3)}\quad ,\qquad -\frac{\pi}{3} < \theta + \frac{2\pi}{3} \leqq \pi$$

and

$$\sqrt{w} = r\, e^{i(\theta/2+\pi/3)}$$

since $\theta + \frac{2\pi}{3}$ is in a proper domain for the principal square root. Thus,

$$\zeta_1 = \frac{2}{3} w^{3/2} = \frac{2}{3} r^{3/2}\, e^{i(3\theta/2+\pi)} = -\zeta_2$$

Therefore $e^{-\xi_1-\xi_2} = e^0 = 1$ and no scaling is necessary.

Case II, $\frac{\pi}{3} < \theta \leqq \pi$.

In this case,

$$\zeta_2 = \frac{2}{3} r^{3/2}\, e^{3\theta i/2}$$

and the principal square root can be used. On the other hand,

$$w = r\, e^{i(\theta + 2\pi/3)} \qquad \pi < \theta + \frac{2\pi}{3} \leqq \frac{5\pi}{3}$$

$$= r\, e^{i(\theta - 4\pi/3)} \qquad -\pi < \theta - \frac{4\pi}{3} \leqq \frac{\pi}{3}$$

and $\theta - 4\pi/3$ is the proper argument for the principal square root. Then,

$$\zeta_1 = \frac{2}{3} r^{3/2}\, e^{i(\theta - 4\pi/3)}\, e^{i(\theta/2 - 2\pi/3)}$$

$$= \frac{2}{3} r^{3/2}\, e^{3\theta i/2} = \zeta_2$$

Thus, the scaling factor for use with KODE = 2 in the argument range $\pi/3 < \theta \leqq \pi$ is

$$e^{-\zeta_1-\zeta_2} = e^{-2\zeta_1} = e^{-(4/3)z^{3/2}} .$$

11. Errata for Reference [1] and Reference [4]

The roots of $Y_0(z)$ and $Y_1(z)$ from CBESY and ZBESY were checked against the roots published in [1, p. 373]. All values checked except the evaluation of $Y_0(z)$ at the third root of $Y_1(z)$. The entry is in error in the 7th digit of the imaginary part. The correct entries are

Zero of Y_1	Y_0
$-7.015903683 + 0.553393046i$	$-0.020126949 + 0.518642\underline{5}33i$.

In [4, p. 220], the 4th zero of $Y_4(z)$ should be

$$-3.\underline{4}307435178 + 1.39457035 62i$$

where the error in [4] is in the first decimal place of the real part.

References

1. Abramowitz, M. and Stegun, I. A., Handbook of Mathematical Functions, NBS Applied Math Series 55, U. S. Dept. of Commerce, 1955.

2. Amos, D. E., Computation of Bessel Functions of Complex Argument, SAND83-0086, May, 1983.

3. Amos, D. E., Computation of Bessel Functions of Complex Argument and Large Order, SAND83-0643, May, 1983.
4. Doring, Boro, Complex Zeros of Cylinder Functions, Math. Comp., 20, 1966, pp. 215-222.

5. Fong, K. W., Jefferson, T. H., Suyehiro, T., Formal Conventions of the SLATEC Library, ACM SIGNUM Newsletter, Vol. 19, No.1, January 1984, pp. 17-22.

6. Fox, P. A., Hall, A. D. and Schryer, N. L., Framework For A Portable Library, ACM Trans. Math. Software, 4, June, 1978, pp. 177-188.

7. Jones, R. E. and Kahaner, D. K., XERROR, The SLATEC Error-Handling Package, SAND82-0800, May, 1982.

Distribution:

1000	W. F. Brinkman
1241	J. P. Quintenz
1541	H. C. Hardee
1600	R. G. Clem
	Attn: 1601 G. W. Kuswa
	1620 R. D. Andreas
	1630 R. C. Maydew
	1650 D. J. Rigali
1601	L. B. Dean
1640	G. J. Simmons
1641	R. J. Thompson
1642	L. F. Shampine
1642	D. E. Amos (35)
1651	W. Williams
1651	J. S. Yu
2322	E. M. Gurrola
2322	D. J. Riley
2343	B. C. Brock
2552	R. D. Moyer
2644	T. E. Koontz
2646	W. H. Vandevender (5)
2646	R. J. Hanson
6240	L. C. Bartel
6241	P. C. Lysne
6256	D. Engi
7553	M. Morris
7553	K. M. Damrau
7553	W. A. Johnson
8024	M. A. Pound
8235	T. H. Jefferson (5)
8341	M. I. Baskis
3141	C. M. Ostrander (5)
3151	W. L. Garner (3)
	For DOE/TIC (Unlim. Release)
3154-3	C. H. Dalin (28)
	For DOE/TIC