NASA Technical Memorandum 105154

51749
p-66

Computer Program for Bessel and Hankel Functions

Kevin L. Kreider
University of Akron
Akron, Ohio

and

Arthur V. Saule, Edward J. Rice, and Bruce J. Clark
Lewis Research Center
Cleveland, Ohio

October 1991

(NASA-TM-105154) COMPUTER PROGRAM FOR BESSEL AND HANKEL FUNCTIONS (NASA) 66 p CSCL 09B
N92-11691
Unclas
G3/61 0051749

NASA

COMPUTER PROGRAM FOR BESSEL AND HANKEL FUNCTIONS

Kevin L. Kreider
University of Akron
Department of Mathematical Sciences
Akron Ohio 44325

Arthur V. Saule*, Edward J. Rice, and Bruce J. Clark
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

E-6439

SUMMARY

Bessel and Hankel functions are widely used in many research and application areas. A set of FORTRAN subroutines for calculating these functions is presented. The routines calculate Bessel and Hankel functions of the first and second kinds, as well as their derivatives, for wide ranges of integer order and real or complex argument in single or double precision. Depending on the order and argument, one of three evaluation methods is used: the power series definition, an Airy function expansion, or an asymptotic expansion. Routines to calculate Airy functions and their derivatives are also included.

1. INTRODUCTION

Since Bessel and Hankel functions are widely used in many research and application areas, it is important to have easy-to-use, reliable computer code to calculate them. The code presented here is designed to satisfy that need, in the form of FORTRAN routines to calculate J_m, Y_m (the Bessel functions of first and second kind) and their first derivatives, as well as the Hankel functions $H^{(1)}_m$ and $H^{(1)}_m$ and their derivatives, for wide ranges of order and argument.

The package consists of a set of FORTRAN routines that calculate Bessel and Hankel functions with real or complex argument in single or double precision. The user writes a calling program that specifies the order m, the argument x, and the function/s to be computed. This input is sent to a driver that calls the appropriate routine to do the calculation.

Three different evaluation methods are used, depending on the combination of input variables: (1) if the order and argument are sufficient small, the power series definition is used directly, (2) if both order and argument are sufficiently large, a series form involving Airy function is used, or (3) if the order is small but the argument is large, an asymptotic expansion is used.

An outline of this report follows: Section 2 contains a description of the use of the routines, including sample programs and a table of output values. Section 3 lists restrictions on input variable ranges. Concluding remarks appear in section 4. The analysis of the three evaluation methods appears in appendix A, while appendix B contains the FORTRAN code listing of the routines.

---

*Retired from NASA Lewis Research Center.

2. USING THE ROUTINES

The available routines for calculating Bessel functions in single precision are RBESSEL (real argument) and CBESSEL (complex argument); for calculating Hankel functions there are RHANKEL (real argument) and CHANKEL (complex argument). The double precision versions of these routines are RDBESSEL, CDBESSEL, RDHANKEL and CDHANKEL, respectively. Each routine examines the input variables, decides which evaluation method to use (see appendix A), calls an appropriate calculation routine to compute the desired value, and returns that value to the calling program through output variables labeled by the user.

In each case, the inputs are:

m	the integer order of the Bessel/Hankel function
x	the real/complex argument of the Bessel/Hankel function
icode	1 means compute the Bessel/Hankel function of the first kind
	2 means compute the Bessel/Hankel function of the second kind
	3 means compute the first and second functions and their derivatives

For each routine, there are four output variables. If icode = 1, only the first variable (besj, han1, etc) is calculated; if icode = 2, only the second variable (besy, han2, etc) is calculated; if icode = 3, all four variables are calculated.

In the sample programs that follow, both single and double precision versions of the routines are called. It is not necessary to declare both single and double precision variables if only one version of the routine is called. However, if icode equals 1 or 2, it is still necessary to include all four output variables in the subroutine call. For the sample programs below, note that different compilers may require slightly different syntax in places; for example, 'double precision' may need to be changed to 'real*8' and 'double complex' to 'complex*16'.

All calculations are done in double precision, but values are returned in either single or double precision depending on the calling routine.

2.1. RBESSEL and RDBESSEL

Here is a sample program illustrating the use of RBESSEL and RDBESSEL:

```
program rbestest
integer m,icode
real besj,besy,dbesj,dbesy,x
double precision desj,desy,ddesj,ddesy,dx
x = 1.0
dx = 1.0d0
m = 1
icode = 3
call rbessel(x,m,icode,besj,besy,dbesj,dbesy)
call rdbessel(dx,m,icode,desj,desy,ddesj,ddesy)
write(6,*) 'besj  = ',besj,desj
write(6,*) 'besy  = ' ,besy,desy
write(6,*) 'dbesj = ',dbesj,ddesj
```

```
      write(6,*) 'dbesy = ',dbesy,ddesy
      end
```

The output is:

```
besj  =    0.4400506     0.4400505857449335
besy  =   -0.7812128    -0.7812128213002896
dbesj =    0.3251471     0.3251471008130330
dbesy =    0.8694698     0.8694697855159653
```

2.2. CBESSEL and CDBESSEL

Here is a sample program illustrating the use of CBESSEL and CDBESSEL:

```
      program cbestest
      integer m,icode
      complex cbesj,cbesy,cdbesj,cdbesy,x
      double complex cdesj,cdesy,cddesj,cddesy,dx
      x = (1.0,1.0)
      dx = (1.0d0,1.0d0)
      m = 1
      icode = 3
      call cbessel(x,m,icode,cbesj,cbesy,cdbesj,cdbesy)
      call cdbessel(dx,m,icode,cdesj,cdesy,cddesj,cddesy)
      write(6,*) 'besj  = ',cbesj,cdesj
      write(6,*) 'besy  = ',cbesy,cdesy
      write(6,*) 'dbesj = ',cdbesj,cddesj
      write(6,*) 'dbesy = ',cdbesy,cddesy
      end
```

The output is:

```
besj  =  (0.6141603,0.3650281)         (0.6141603082191699,0.3650280509978571)
besy  =  (-0.6576946,0.6298010)        (-0.6576945629749859,0.6298010000344907)
dbesj =  (0.4480143,-0.3719638)        (0.4480143229256401,-0.3719637789376947)
dbesy =  (0.4594212,6.6410817E-02)     (0.4594212124350948,6.6410818437236729E-02)
```

2.3. RHANKEL and RDHANKEL

Here is a sample program illustrating the use of RHANKEL and RDHANKEL:

```
      program rhantest
      integer m,icode
      complex han1,han2,dhan1,dhan2,x
      double complex dan1,dan2,ddan1,ddan2,dx
      x = 1.0
      dx = 1.0d0
      m = 1
```

```
icode = 3
call rhankel(x,m,icode,han1,han2,dhan1,dhan2)
call rdhankel(dx,m,icode,dan1,dan2,ddan1,ddan2)
write(6,*) 'han1  = ',han1,dan1
write(6,*) 'han2  = ',han2,dan2
write(6,*) 'dhan1 = ',dhan1,ddan1
write(6,*) 'dhan2 = ',dhan2,ddan2
end
```

The output is:

```
han1  =  (0.4400506,-0.7812128)     (0.4400505857449335,-0.7812128213002896)
han2  =  (0.4400506,0.7812128)      (0.4400505857449335,0.7812128213002896)
dhan1 =  (0.3251471,0.8694698)      (0.3251471008130330,0.8694697855159653)
dhan2 =  (0.3251471,-0.8694698)     (0.3251471008130330,-0.8694697855159653)
```

2.4. CHANKEL and CDHANKEL

Here is a sample program illustrating the use of CHANKEL and CDHANKEL:

```
program chantest
integer m,icode
complex chan1,chan2,cdhan1,cdhan2,x
double complex cdan1,cdan2,cddan1,cddan2,dx
x = (1.0,1.0)
dx = (1.0d0,1.0d0)
m = 1
icode = 3
call chankel(x,m,icode,chan1,chan2,cdhan1,cdhan2)
call cdhankel(dx,m,icode,cdan1,cdan2,cddan1,cddan2)
write(6,*) 'han1  = ',chan1,cdan1
write(6,*) 'han2  = ',chan2,cdan2
write(6,*) 'dhan1 = ',cdhan1,cddan1
write(6,*) 'dhan2 = ',cdhan2,cddan2
end
```

The output is:

```
han1  =  (-1.5640676E-02,-0.2926665)   (-1.5640691815320795E-02,-0.2926665119771287)
han2  =  (1.243961,1.022723)           (1.243961308253660,1.022722613972843)
dhan1 =  (0.3816035,8.7457448E-02)     (0.3816035044884034,8.7457433497400103E-02)
dhan2 =  (0.5144252,-0.8313850)        (0.5144251413628769,-0.8313849913727896)
```

Tables I to IV list Bessel function values at various points. Each of the three evaluation methods is represented in the tables. These values were generated on a Unix-based workstation using the f77 compiler. Different compilers may give slightly different results.

3. CONDITIONS ON INPUT VARIABLES

The following restrictions apply to the inputs:

(1) The order m must be a nonnegative integer.
(2) The argument must be a nonnegative real number x, or a complex number z.
(3) Avoid parameter values over 32 000 in single precision.
(4) If $x \leq 2(\text{ERTOLM}*m!)\hat{}(1/m)$, the Bessel functions are set to 0 or ERTOLP (machine infinity, defined below), as appropriate, due to underflow in the power series and Airy function evaluations. For complex argument, replace x by $|z|$.
(5) As order and argument both grow, the Airy function calculation breaks down due to overflow. The exact region in which this occurs is difficult to describe, but if the returned function values are 0.000E+00 or about ERTOLP (see below), then it should be assumed that underflow or overflow has occurred.

The machine tolerance limits ERTOLP and ERTOLM are set at the beginning of the routines RBESSEL, CBESSEL, RDBESSEL and CDBESSEL. ERTOLP and ERTOLM are the largest and smallest numbers that don't cause over- or underflow in single precision. Their default values are $1.0\text{x}10^{35}$ and $1.0\text{x}10^{-35}$, and can be easily changed in the source code to accommodate different computers.

4. CONCLUSION

The FORTRAN routines described here calculate the Bessel and Hankel functionsand their first derivatives for integer order and real or complex argument insingle or double precision. One of three different evaluation methods is used,depending on the order and argument: the power series, an Airy functionexpansion, or Hankel's asymptotic expansion.

The original single precision, real argument Bessel routines were written by Art Saule. These were improved and extended to the other cases by Kevin Kreider. Bruce Clark ran numerical tests and suggested further improvements.

APPENDIX A

THE EVALUATION METHODS

A.1. Choosing the Evaluation Method

The three evaluation methods (power series, Airy functions, and asymptotic expansions) are valid in overlapping regions in the (m,x) plane. The driver routines choose which method to use according to the scheme depicted in figures 1 and 2. The difference between the regions in the two figures is due to the behavior of the different methods for complex argument off the real axis.

The boundaries for these zones were established by comparing output values of the three methods with published results (ref. 1) for a wide range of input values. Since there is considerable overlap in the regions in which the three methods are valid, the final placement of the boundaries was guided by judicious interpretation of the data, and is somewhat arbitrary.

A.2. Series Truncation

The three methods used here each involve infinite series, which must betruncated so as to provide accuracy to about 14 digits. For programming purposes, these series are put in nested form by repeatedly factoring outcommon expressions in the terms. This forces a reverse order summation with the final nested term as the starting value, so that the number of terms needed to obtain a sufficiently accurate approximation to the series must be known a priori. For each series, an index (labeled L_{ij} for each series in sections A.3 to A.5) indicating the proper number of terms to be used is developed byexamining the original series: terms are included until the next additional term does not change the desired accuracy of the sum.

A.3. Evaluation Method 1: Power Series

The power series expressions for the Bessel functions are useful for calculation if the order and argument are both relatively small. Figures 1 and 2 indicate the region in which the power series are used for real and complex argument.

As described in section A.2, all series in the code are written in nested form, and infinite series are truncated. The expressions used appear below.

A.3.1. Bessel Functions of the First Kind

From reference 1 (eq. (9.1.10)),

$$J_m(z) = \sum_{k=0}^{\infty} \frac{(-1)^k (z/2)^{2k+m}}{k!(k+m)!}$$
$$\approx \frac{(z/2)^m}{m!}\left[1 - A_{m1}\left(\frac{z}{2}\right)^2\left[1 - A_{m2}\left(\frac{z}{2}\right)^2\left[1 - \cdots\left[1 - A_{mL_{11}}\left(\frac{z}{2}\right)^2\right]\right]\right]\right] \qquad (3.1)$$

$$A_{mk} = \frac{1}{k(m + k)}, \qquad k = 1, 2, \ldots, L_{11} \tag{3.2}$$

$$L_{11} = \text{int}\left(10 + \frac{4}{3}|z|\right) \tag{3.3}$$

Differentiating (3.1) gives

$$J_m'(z) = \sum_{k=0}^{\infty} \frac{(-1)^k(2k + m)(z/2)^{2k+m-1}}{2k!(k + m)!}$$
$$\approx \frac{(z/2)^{m-1}}{2(m-1)!}\left[1 - A'_{m1}\left(\frac{z}{2}\right)^2\left[1 - A'_{m2}\left(\frac{z}{2}\right)^2\left[1 - \ldots \left[1 - A'_{mL_{12}}\left(\frac{z}{2}\right)^2\right]\right]\right]\right] \tag{3.4}$$

$$A'_{mk} = \frac{m + 2k}{k(m + k)(m + 2k - 2)}, \qquad k = 1, 2, \ldots, L_{12} \tag{3.5}$$

$$L_{12} = \text{int}(10 + 1.6|z|) \tag{3.6}$$

A.3.2. Bessel Functions of the Second Kind

From reference 2 (eq. (77), sec 4.8) (a slightly modified version appears in ref. 1 (eq. (9.1.11)),

$$Y_m(z) = \left(\frac{2}{\pi}\right)J_m(z)\left[\ln\left(\frac{z}{2}\right) + \gamma\right] + G_m(z) + F_m(z) \tag{3.7}$$

$$\varphi(0) = 0 \text{ (digamma function)}$$

$$\varphi(n) = \sum_{j=1}^{n} \frac{1}{j}, \qquad n \geq 1 \tag{3.8}$$

$$\gamma = 0.5772\ 1566\ 49015 \text{ (Euler's constant)} \tag{3.9}$$

$$G_m(z) = -\frac{1}{\pi}\sum_{k=0}^{\infty}\frac{(-1)^k(z/2)^{2k+m}[\varphi(k) + \varphi(m + k)]}{k!(m + k)!}$$

$$= -\frac{(z/2)^m}{\pi m!}\left[\varphi(m) - \left(\frac{z}{2}\right)^2\left(\frac{1 + \varphi(m + 1)}{m + 1}\right)S_m(z)\right] \tag{3.10}$$

$$S_m(z) \approx 1 - D_{m1}\left(\frac{z}{2}\right)^2\left[1 - D_{m2}\left(\frac{z}{2}\right)^2\left[1 - \ldots\left[1 - D_{mL_{13}}\left(\frac{z}{2}\right)^2\right]\right]\right] \tag{3.11}$$

$$D_{mk} = \frac{\varphi(k + 1) + \varphi(m + k + 1)}{[\varphi(k) + \varphi(m + k)](k + 1)(m + k + 1)},$$

$$k = 1, 2, \ldots, L_{13} \qquad m = 0, 1, 2, \ldots \tag{3.12}$$

$$L_{13} = \text{int}(7 + 1.5|z|). \tag{3.13}$$

For $m > 1$,

$$F_m(z) = -\frac{1}{\pi}\sum_{k=0}^{m-1}\frac{(z/2)^{2k-m}(m - k - 1)!}{k!}.$$

$$= -\frac{(m - 1)!}{\pi(z/2)^m}\left[1 + B_{m1}\left(\frac{z}{2}\right)^2\left[1 + \ldots\left[1 + B_{m,m-1}\left(\frac{z}{2}\right)^2\right]\right]\right] \tag{3.14}$$

$$B_{mk} = \frac{1}{k(m - k)}, \qquad k = 1, 2, \ldots, m - 1 \tag{3.15}$$

$$F_0(z) = 0, \qquad F_1(z) = -\frac{2}{\pi z} \tag{3.16}$$

Differentiating (3.7) gives

$$Y_m'(z) = \left(\frac{2}{\pi}\right)J_m'(z)\left[\ln\left(\frac{z}{2}\right) + \gamma\right] + \frac{2}{\pi z}J_m(z) + G_m'(z) + F_m'(z) \tag{3.17}$$

$$G_m'(z) = -\frac{1}{2\pi}\sum_{k=0}^{\infty}\frac{(-1)^k\left[\varphi(k)+\varphi(k+m)\right](m+2k)(z/2)^{m+2k-1}}{k!(m+k)!}$$

$$= -\frac{(z/2)^{m-1}}{2\pi m!}\left[m\varphi(m) - \frac{\left[1+\varphi(m+1)\right](m+2)(z/2)^2}{m+1}S_m'(z)\right] \qquad (3.18)$$

$$S_m'(z) \approx 1 - D_{m1}'\left(\frac{z}{2}\right)^2\left[1 - D_{m2}'\left(\frac{z}{2}\right)^2\left[1 - \ldots\left[1 - D_{mL_{14}}'\left(\frac{z}{2}\right)^2\right]\right]\right] \qquad (3.19)$$

$$D_{mk}' = \frac{\left[\varphi(k+1)+\varphi(m+k+1)\right](m+2k+2)}{\left[\varphi(k)+\varphi(m+k)\right](m+k+1)(k+1)(m+2k)}, \qquad k = 1, 2, \ldots, L_{14} \qquad (3.20)$$

$$L_{14} = \mathrm{int}(8 + 1.4|z|). \qquad (3.21)$$

For $m > 1$,

$$F_m'(z) = \frac{1}{2\pi}\sum_{k=0}^{m-1}\frac{(m-2k)(m-k-1)!(z/2)^{2k-m-1}}{k!}$$

$$= \frac{1}{2\pi}\left[\frac{m!/(z/2)^m}{z/2} + \sum_{k=1}^{m-1}\left\{\frac{(m-k-1)!}{(z/2)^{m-k-1}}\right\}\left(\frac{(z/2)^k}{k!}\right)\left\{\frac{m-2k}{(z/2)^2}\right\}\right] \qquad (3.22)$$

(This form of F_m' is chosen since it is convenient for programming using the function routines RDFACT and CDFACT.)

$$F_0'(z) = 0, \qquad F_1'(z) = \frac{2}{\pi z^2}. \qquad (3.23)$$

A.4. Evaluation Method 2: Airy Function Approximation

Calculating Bessel functions with relatively large order and argument can be done with uniformly asymptotic expressions involving Airy functions. Figures 1 and 2 indicate the region in which these expressions are used for real and complex argument. Calculation of the Airy functions is discussed in section 10.

As described in section A.2, all series in the code are written in nested form, and infinite series are truncated. The expressions used appear below.

A.4.1. Bessel Functions of the First Kind

From reference 1 (eq. (9.3.3)),

$$J_m(mz) \sim \left(\frac{4\zeta}{1 - z^2}\right)^{1/4}\left[\frac{A_i\left(m^{2/3}\zeta\right)}{m^{1/3}}\sum_{k=0}^{\infty}\frac{a_k(\zeta)}{m^{2k}} + \frac{A_i'\left(m^{2/3}\zeta\right)}{m^{5/3}}\sum_{k=0}^{\infty}\frac{b_k(\zeta)}{m^{2k}}\right]. \tag{4.1}$$

Several comments must be made here. First, from reference 1 (eqs. (9.3.38) and (9.3.39)),

$$\text{If } \operatorname{Re}(z) \geq 1, \ \frac{2}{3}(-\zeta)^{3/2} = \sqrt{z^2 - 1} - \arccos\left(\frac{1}{z}\right) \tag{4.2}$$

$$\text{If } \operatorname{Re}(z) < 1, \ \frac{2}{3}(\zeta)^{3/2} = \ln\left(\frac{1 + \sqrt{1 - z^2}}{z}\right) - \sqrt{1 - z^2} \tag{4.3}$$

(remember, Re(z) is assumed to be nonnegative). Second, $(4\zeta/1 - z)$ becomes indeterminate as $z \rightarrow 1$, so for $|z - 1| < 0.02$,

$$\left(\frac{4\zeta}{1 - z^2}\right)^{1/4} \approx 2^{1/3}\left(1 + \frac{h}{5} + \frac{3h^2}{35} + \frac{73h^3}{1575} + \frac{35\ 209h^4}{1\ 212\ 750} + \frac{380\ 069h^5}{18\ 768\ 750}\right) \tag{4.4}$$

$$h = 1 - z \tag{4.5}$$

Finally, the coefficients a and b (ref. 1, eq. (9.3.40)) are so small that only a_0, a_1, a_2, b_0, b_1, and b_2 are needed to obtain the desired accuracy. They are:

$$a_0(\zeta) = 1 \tag{4.6}$$

$$a_1(\zeta) = u_2 - \frac{7\gamma_1}{48\zeta} - \frac{455}{4608\zeta^3} \tag{4.7}$$

$$a_2(\zeta) = u_4 - \frac{7}{48\zeta}\left[\gamma_3 + \frac{65}{96\zeta^2}\left\{u_2 + \frac{209}{144\zeta}\left(\gamma_1 + \frac{425}{192\zeta^2}\right)\right\}\right] \tag{4.8}$$

$$b_0(\zeta) = -\gamma_1 - \frac{5}{48\,\zeta^2} \tag{4.9}$$

$$b_1(\zeta) = -\gamma_3 - \frac{5}{48\,\zeta^2}\left[u_2 + \frac{77}{96\,\zeta}\left\{\gamma_1 + \frac{221}{144\,\zeta^2}\right\}\right] \tag{4.10}$$

$$b_2(\zeta) = -\gamma_5 - \frac{5}{48\,\zeta^2}\left[u_4 + \frac{77}{96\,\zeta}\left\{\gamma_3 + \frac{221}{144\,\zeta^2}\left(u_2 + \frac{437}{192\,\zeta}\left[\gamma_1 + \frac{145}{48\,\zeta^2}\right]\right)\right\}\right] \tag{4.11}$$

where auxilliary coefficients are given by

$$\nu = 1 - z^2, \qquad \xi = \sqrt{\frac{1 - z^2}{\zeta}}$$

$$\gamma_1 = u_1\,\zeta^{-1/2} = \frac{\xi}{8\nu}\left(1 - \frac{5}{3\nu}\right)$$

$$u_2 = \frac{1}{64\nu}\left(\frac{9}{2} - \frac{77}{3\nu}\left(1 - \frac{5}{6\nu}\right)\right)$$

$$\gamma_3 = u_3\,\zeta^{-1/2} = \frac{\xi}{512\nu^2}\left[\frac{75}{2} - \frac{4563}{10\nu} + \frac{17\,017}{18\nu^2}\left(1 - \frac{5}{9\nu}\right)\right]$$

$$u_4 = \frac{1}{4096\nu^2}\left[\frac{3675}{8} - \frac{96\,833}{10\nu} + \frac{2717}{\nu^2}\left\{\frac{53}{4} - \frac{2737}{162\nu}\left(1 - \frac{5}{12\nu}\right)\right\}\right]$$

$$\gamma_5 = u_5\,\zeta^{-1/2} = \frac{\xi}{32\,768\nu^3}\left[\frac{59\,535}{8} - \frac{221}{4\nu}\left\{\frac{305\,923}{70} - \frac{77}{\nu}\left(\frac{14\,743}{45} - \frac{95}{\nu}\left\{\frac{67}{9} - \frac{3335}{486\nu}\left(1 - \frac{1}{3\nu}\right)\right\}\right)\right\}\right]$$

The approximate expression for $J_m(mz)$ is then

$$J_m(mz) \approx \left(\frac{4\zeta}{1 - z^2}\right)^{1/4} \left[\frac{A_i\left(m^{2/3}\zeta\right)}{m^{1/3}} \left\{1 + \frac{a_1(\zeta)}{m^2} + \frac{a_2(\zeta)}{m^4}\right\} + \frac{A_i'\left(m^{2/3}\zeta\right)}{m^{5/3}} \left\{b_0(\zeta) + \frac{b_1(\zeta)}{m^2} + \frac{b_2(\zeta)}{m^4}\right\}\right] \tag{4.12}$$

The analogous expression for $J_m'(mz)$ is (ref. 1, eq. (9.3.43))

$$\begin{aligned} J_m'(mz) &\sim -\frac{2}{z}\left(\frac{1 - z^2}{4\zeta}\right)^{1/4} \left[\frac{A_i\left(m^{2/3}\zeta\right)}{m^{4/3}} \sum_{k=0}^{\infty} \frac{c_k(\zeta)}{m^{2k}} + \frac{A_i'\left(m^{2/3}\zeta\right)}{m^{2/3}} \sum_{k=0}^{\infty} \frac{d_k(\zeta)}{m^{2k}}\right] \\ &\approx -\frac{2}{z}\left(\frac{1 - z^2}{4\zeta}\right)^{1/4} \left[\frac{A_i\left(m^{2/3}\zeta\right)}{m^{4/3}} c_0(\zeta) + \frac{A_i'\left(m^{2/3}\zeta\right)}{m^{2/3}} \left(1 + \frac{d_1(\zeta)}{m^2}\right)\right] \end{aligned} \tag{4.13}$$

$$c_0(\zeta) = \frac{7}{48\zeta} - \frac{7}{24\zeta^{1/2}}\left(1 - z^2\right)^{-3/2} + \frac{3}{8\zeta^{1/2}}\left(1 - z^2\right)^{-1/2} \tag{4.14}$$

$$d_0(\zeta) = 1 \tag{4.15}$$

$$d_1(\zeta) = \frac{385}{4608}\zeta^{-3} + \frac{5}{128}\left(\zeta\left(1 - z^2\right)\right)^{-3/2}\left(\frac{7}{9} - \left(1 - z^2\right)\right) - \frac{455}{1152}\left(1 - z^2\right)^{-3} + \frac{33}{64}\left(1 - z^2\right)^{-2} - \frac{15}{128}\left(1 - z^2\right)^{-1} \tag{4.16}$$

A.4.2. Bessel Functions of the Second Kind

From reference 1 (eq. (9.2.36))

$$Y_m(mz) \sim -\left(\frac{4\zeta}{1-z^2}\right)^{1/4}\left[\frac{B_i(m^{2/3}\zeta)}{m^{1/3}}\sum_{k=0}^{\infty}\frac{a_k(\zeta)}{m^{2k}} + \frac{B_i'(m^{2/3}\zeta)}{m^{5/3}}\sum_{k=0}^{\infty}\frac{b_k(\zeta)}{m^{2k}}\right]$$

$$\approx -\left(\frac{4\zeta}{1-z^2}\right)^{1/4}\left[\frac{B_i(m^{2/3}\zeta)}{m^{1/3}}\left\{1 + \frac{a_1(\zeta)}{m^2} + \frac{a_2(\zeta)}{m^4}\right\} + \frac{B_i'(m^{2/3}\zeta)}{m^{5/3}}\left\{b_0(\zeta) + \frac{b_1(\zeta)}{m^2} + \frac{b_2(\zeta)}{m^4}\right\}\right] \tag{4.17}$$

From reference 1 (eq. (9.3.44))

$$Y_m'(mz) \sim \frac{2}{z}\left(\frac{1-z^2}{4\zeta}\right)^{1/4}\left[\frac{B_i(m^{2/3}\zeta)}{m^{4/3}}\sum_{k=0}^{\infty}\frac{c_k(\zeta)}{m^{2k}} + \frac{B_i'(m^{2/3}\zeta)}{m^{2/3}}\sum_{k=0}^{\infty}\frac{d_k(\zeta)}{m^{2k}}\right]$$

$$\approx \frac{2}{z}\left(\frac{1-z^2}{4\zeta}\right)^{1/4}\left[\frac{B_i(m^{2/3}\zeta)}{m^{4/3}}c_0(\zeta) + \frac{B_i'(m^{2/3}\zeta)}{m^{2/3}}\left\{1 + \frac{d_1(\zeta)}{m^2}\right\}\right] \tag{4.18}$$

A.5. Evaluation Method 3: Hankel's Asymptotic Expansion

Hankel's asymptotic expansions can be used to approximate Bessel functions if the order is small and the argument large. Figures 1 and 2 indicate the region in which these expansions are used for real and complex argument.

As described in section A.2, all series in the code are written in nested form, and infinite series are truncated. The expressions used appear below.

A.5.1. Bessel Functions of the First Kind

From reference 1 (eq. (9.2.5))

$$J_m(z) = \sqrt{\frac{2}{\pi z}}\left[P_m(z)\cos\chi - Q_m(z)\sin\chi\right] \tag{5.1}$$

$$\chi = z - \left(\frac{m}{2} + \frac{1}{4}\right)\pi \tag{5.2}$$

$$\begin{aligned} P_m(z) &\sim \sum_{k=0}^{\infty} \frac{(-1)^k (m,2k)}{(2z)^{2k}} \\ &\approx 1 - \frac{A_1}{(8z)^2}\left[1 - \frac{A_2}{(8z)^2}\left[1 - \ldots \left[1 - \frac{A_{L_{31}}}{(8z)^2}\right]\right]\right] \end{aligned} \tag{5.3}$$

$$A_k = \frac{4\left(4m^2 - (4k-3)^2\right)\left(4m^2 - (4k-1)^2\right)}{4k(4k-2)}, \qquad k = 1, 2, \ldots, L_{31} \tag{5.4}$$

$$L_{31} = \mathrm{int}\left(2 + \frac{m}{3} + \frac{27}{\sqrt{m - 12}}\right) \tag{5.5}$$

$$\begin{aligned} Q_m(z) &\sim \sum_{k=0}^{\infty} \frac{(-1)^k (m,2k+1)}{(2z)^{2k+1}} \\ &\approx \frac{4m^2 - 1}{8z}\left[1 - \frac{B_1}{(8z)^2}\left[1 - \frac{B_2}{(8z)^2}\left[1 - \ldots \left[1 - \frac{B_{L_{31}}}{(8z)^2}\right]\right]\right]\right] \end{aligned} \tag{5.6}$$

$$B_k = \frac{4\left(4m^2 - (4k-1)^2\right)\left(4m^2 - (4k+1)^2\right)}{4k(4k+2)}, \qquad k = 1, 2, \ldots, L_{31} \tag{5.7}$$

$$(m,n) = \frac{\Gamma\left(\frac{1}{2} + m + n\right)}{\Gamma\left(\frac{1}{2} + m - n\right) n!} \tag{5.8}$$

The derivative is given by reference 1 (eq. (9.2.11))

$$J_m'(z) = \sqrt{\frac{2}{\pi z}} \left(-R_m(z)\sin \chi - S_m(z)\cos \chi\right) \tag{5.9}$$

$$\begin{aligned} R_m(z) &\sim \sum_{k=0}^{\infty} (-1)^k \left(\frac{4m^2 + 16k^2 - 1}{4m^2 - (4k - 1)^2}\right) \frac{(m,2k)}{(2k)^{2k}} \\ &\approx 1 - \frac{A_1'}{(8z)^2}\left[1 - \frac{A_2'}{(8z)^2}\left[1 - \ldots \left[1 - \frac{A_{L_{31}}'}{(8z)^2}\right]\right]\right] \end{aligned} \tag{5.10}$$

$$A_k' = \frac{4\left(4m^2 + (4k)^2 - 1\right)\left(4m^2 - (4k - 3)^2\right)\left(4m^2 - (4k - 5)^2\right)}{4k(4k - 2)\left(4m^2 + (4k - 4)^2 - 1\right)}, \qquad k = 1, 2, \ldots, L_{31} \tag{5.11}$$

$$\begin{aligned} S_m(z) &\sim \sum_{k=0}^{\infty} (-1)^k \left(\frac{4m^2 + 4(2k + 1)^2 - 1}{4m^2 - (4k + 1)^2}\right) \frac{(m,2k + 1)}{(2z)^{2k+1}} \\ &\approx \frac{4m^2 + 3}{8z}\left[1 - \frac{B_1'}{(8z)^2}\left[1 - \ldots \left[1 - \frac{B_{L_{31}}'}{(8z)^2}\right]\right]\right] \end{aligned} \tag{5.12}$$

$$B_k' = \frac{4\left(4m^2 + (4k + 2)^2 - 1\right)\left(4m^2 - (4k - 3)^2\right)\left(4m^2 - (4k - 1)^2\right)}{4k(4k + 2)\left(4m^2 + (4k - 2)^2 - 1\right)}, \qquad k = 1, 2, \ldots, L_{31} \tag{5.13}$$

A.5.2. Bessel Functions of the Second Kind

From reference 1 (eqs. (9.2.6) and (9.2.12)),

$$Y_m(z) \sim \sqrt{\frac{2}{\pi z}} \left[P_m(z)\cos \chi + Q_m(z)\sin \chi\right] \tag{5.14}$$

$$Y_m'(z) \sim \sqrt{\frac{2}{\pi z}} \left[R_m(z)\cos\chi - S_m(z)\sin\chi\right] \tag{5.15}$$

A.6. Airy Functions

Three different evaluation methods are used to calculate the Airy functions Ai(y) and Bi(y) and their derivatives, depending on $y = re^{i\theta}$. For r sufficiently small, the power series is used. Otherwise, asymptotic expansions are used - an exponential form if $\theta \epsilon \left[0, \frac{2\pi}{3}\right] \cup \left[\frac{4\pi}{3}, 2\pi\right]$, or a trigonometric form. Figures 3 and 4 indicate the regions in which these different methods are used for real and complex argument.

As described in section A.2, all series in the code are written in nested form, and infinite series are truncated. The expressions used appear below.

A.6.1. Power Series

From reference 1 (eqs. (10.4.2) to (10.4.5)),

$$Ai(y) = c_1 f(y) - c_2 g(y) \tag{6.1}$$

$$Bi(y) = \sqrt{3}\left(c_1 f(y) + c_2 g(y)\right) \tag{6.2}$$

$$c_1 = 0.35502\ 80538\ 87817 \tag{6.3}$$

$$c_2 = 0.25881\ 94037\ 92807 \tag{6.4}$$

$$f(y) = \sum_{k=0}^{\infty} 3^k \left(\frac{1}{3}\right)_k \frac{y^{3k}}{(3k)!}$$

$$\approx 1 + F_1 y^3 \left[1 + F_2 y^3 \left[1 + \ldots \left[1 + F_{L_{25}} y^3\right]\right]\right] \tag{6.5}$$

$$F_k = \frac{1}{3k(3k-1)}, \qquad k = 1, 2, \ldots, L_{25} \tag{6.6}$$

$$g(y) = \sum_{k=0}^{\infty} 3^k \left(\frac{2}{3}\right)_k \frac{y^{3k+1}}{(3k + 1)!}$$

$$\approx y\left[1 + G_1 y^3\left[1 + G_2 y^3\left[1 + \ldots + \left[1 + G_{L_{25}} y^3\right]\right]\right]\right] \tag{6.7}$$

$$G_k = \frac{1}{3k(3k + 1)}, \qquad k = 1, 2, \ldots, L_{25} \tag{6.8}$$

$$L_{25} = \mathrm{int}\left(5 + 3.5|y|\right) \tag{6.9}$$

$$3^k\left(\alpha + \frac{1}{3}\right)_k = (3\alpha + 1)(3\alpha + 4) \ldots (3\alpha + 3k - 2) \tag{6.10}$$

The power series has limited usefulness in the complex domain due to cancellation error caused by the subtraction in (6.1), hence the limiting radius is lower than for strictly real argument.

For the derivatives,

$$Ai'(y) = c_1 f'(y) - c_2 g'(y) \tag{6.11}$$

$$Bi'(y) = \sqrt{3}\left(c_1 f'(y) + c_2 g'(y)\right) \tag{6.12}$$

$$f'(y) = \sum_{k=0}^{\infty} 3^k \left(\frac{1}{3}\right)_k \frac{y^{3k-1}}{(3k - 1)!}$$

$$\approx \frac{1}{2} y^2\left[1 + F'_1 y^3\left[1 + \ldots \left[1 + F'_{L_{25}} y^3\right]\right]\right] \tag{6.13}$$

$$F'_k = \frac{1}{3k(3k + 2)}, \qquad k = 1, 2, \ldots, L_{25} \tag{6.14}$$

$$g'(y) = \sum_{k=0}^{\infty} 3^k \left(\frac{2}{3}\right)_k \frac{y^{3k}}{(3k)!} \tag{6.15}$$

$$\approx 1 + G_1' y^3\left[1 + G_2' y^3\left[1 + \ldots \left[1 + G_{L_{25}}' y^3\right]\right]\right]$$

$$G_k' = \frac{1}{3k(3k - 2)}, \qquad k = 1, 2, \ldots, L_{25} \tag{6.16}$$

A.6.2. Trigonometric Asymptotic Expansion

From reference 1 (eqs. (10.4.60) and (10.4.64))

$$Ai(-y) \sim \pi^{-1/2} y^{-1/4}\left[\sin\left(\lambda + \frac{\pi}{4}\right) S_{21}(\lambda) - \cos\left(\lambda + \frac{\pi}{4}\right) S_{22}(\lambda)\right] \tag{6.17}$$

$$Bi(-y) \sim \pi^{-1/2} y^{-1/4}\left[\cos\left(\lambda + \frac{\pi}{4}\right) S_{21}(\lambda) + \sin\left(\lambda + \frac{\pi}{4}\right) S_{22}(\lambda)\right] \tag{6.18}$$

$$\lambda = \frac{2}{3} y^{3/2} \tag{6.19}$$

$$c_k = \frac{(2k + 1)(2k + 3) \ldots (6k - 1)}{216^k k!}, \qquad c_0 = 1 \tag{6.20}$$

$$S_{21}(\lambda) = \sum_{k=0}^{\infty} (-1)^k c_{2k} \lambda^{-2k} \tag{6.21}$$

$$\approx 1 - V_1 \lambda^{-2}\left[1 - V_2 \lambda^{-2}\left[1 - \ldots \left[1 - V_{L_{21}} \lambda^{-2}\right]\right]\right]$$

$$V_k = \frac{(12k - 11)(12k - 7)(12k - 5)(12k - 1)}{12^3 k(12k - 6)}, \qquad k = 1, 2, \ldots L_{21} \tag{6.22}$$

$$L_{21} = \text{int}\left(2 + 160\lambda^{-1}\right) \tag{6.23}$$

$$\begin{aligned} S_{22}(\lambda) &= \sum_{k=0}^{\infty} (-1)^k c_{2k+1} \lambda^{-2k-1} \\ &\approx -\frac{5}{72\lambda}\left[1 - W_1\lambda^{-2}\left[1 - W_2\lambda^{-2}\left[1 - \ldots \left[1 - W_{L_{22}}\lambda^{-2}\right]\right]\right]\right] \end{aligned} \tag{6.24}$$

$$W_k = \frac{(12k - 5)(12k - 1)(12k + 1)(12k + 5)}{12^3 k(12k + 6)}, \qquad k = 1, 2, \ldots, L_{22} \tag{6.25}$$

$$L_{22} = \text{int}\left(2 + 220\lambda^{-1}\right) \tag{6.26}$$

For the derivatives, use reference 1 (eqs. (10.4.62) and (10.4.67)),

$$Ai'(-y) \sim \pi^{-1/2} y^{1/4}\left[\cos\left(\lambda + \frac{\pi}{4}\right) S_{23}(\lambda) + \sin\left(\lambda + \frac{\pi}{4}\right) S_{24}(\lambda)\right] \tag{6.27}$$

$$Bi'(-y) \sim \pi^{-1/2} y^{1/4}\left[\sin\left(\lambda + \frac{\pi}{4}\right) S_{23}(\lambda) - \cos\left(\lambda + \frac{\pi}{4}\right) S_{24}(\lambda)\right] \tag{6.28}$$

$$\begin{aligned} S_{23}(\lambda) &= \sum_{k=0}^{\infty} (-1)^k d_{2k} \lambda^{-2k} \\ &\approx 1 - V_1' \lambda^{-2}\left[1 - V_2' \lambda^{-2}\left[1 - \ldots \left[1 - V_{L_{23}}'\lambda^{-2}\right]\right]\right] \end{aligned} \tag{6.29}$$

$$d_k = -\frac{6k + 1}{6k - 1} c_k, \qquad d_o = 1 \tag{6.30}$$

$$V_k' = \frac{(12k - 13)(12k - 7)(12k - 5)(12k + 1)}{12^3 k(12k - 6)}, \qquad k = 1, 2, \ldots, L_{23} \tag{6.31}$$

$$L_{23} = \text{int}\left(2 + 130\lambda^{-1}\right) \tag{6.32}$$

$$S_{24}(\lambda) = \sum_{k=0}^{\infty} (-1)^k d_{2k+1} \lambda^{-2k-1}$$

$$\approx 1 - W_1' \lambda^{-2}\left[1 - W_2' \lambda^{-2}\left[1 - \ldots \left[1 - W_{L_{24}}' \lambda^{-2}\right]\right]\right] \tag{6.33}$$

$$W_k' = \frac{(12k - 7)(12k - 1)(12k + 1)(12k + 7)}{12^3 k(12k + 6)}, \qquad k = 1, 2, \ldots, L_{24} \tag{6.34}$$

$$L_{24} = \text{int}\left(2 + 200\lambda^{-1}\right) \tag{6.35}$$

A.6.3. Exponential Approximation

From reference 1 (eqs. (10.4.59) and (10.4.63)),

$$\text{Ai}(y) \sim \frac{1}{2} \pi^{-1/2} y^{-1/4} e^{-\lambda} S_1(\lambda) \tag{6.36}$$

$$\text{Bi}(y) \sim \pi^{-1/2} y^{-1/4} e^{\lambda} S_2(\lambda) \tag{6.37}$$

$$S_1(\lambda) = \sum_{k=0}^{\infty} (-1)^k c_k \lambda^{-k}$$

$$\approx 1 - H_1\lambda^{-1}\left[1 - H_2\lambda^{-1}\left[1 - \ldots \left[1 - H_{L_{26}}\lambda^{-1}\right]\right]\right] \tag{6.38}$$

$$S_2(\lambda) = \sum_{k=0}^{\infty} c_k \lambda^{-k}$$

$$\approx 1 + H_1\lambda^{-1}\left[1 + H_2\lambda^{-1}\left[1 + \ldots \left[1 + H_{L_{26}}\lambda^{-1}\right]\right]\right] \tag{6.39}$$

$$H_k = \frac{(6k - 5)(6k - 1)}{72k}, \qquad k = 1, 2, \ldots, L_{26} \tag{6.40}$$

$$L_{26} = \text{int}\left(4 + 300\lambda^{-1}\right) \tag{6.41}$$

For the derivatives, use reference 1 (eqs. (10.4.61) and (10.4.66)),

$$\mathrm{Ai}'(y) \sim -\frac{1}{2}\,\pi^{-1/2}y^{1/4}e^{-\lambda}S_1'(\lambda) \tag{6.42}$$

$$\mathrm{Bi}'(y) \sim \pi^{-1/2}y^{1/4}e^{\lambda}S_2'(\lambda) \tag{6.43}$$

$$\begin{aligned} S_1'(\lambda) &= \sum_{k=0}^{\infty} (-1)^k d_k \lambda^{-k} \\ &\approx 1 - H_1'\,\lambda^{-1}\left[1 - H_2'\,\lambda^{-1}\left[1 - \ldots \left[1 - H_{L_{27}}'\lambda^{-1}\right]\right]\right] \end{aligned} \tag{6.44}$$

$$\begin{aligned} S_2'(\lambda) &= \sum_{k=0}^{\infty} d_k \lambda^{-k} \\ &\approx 1 + H_1'\,\lambda^{-1}\left[1 + H_2'\,\lambda^{-1}\left[1 + \ldots \left[1 + H_{L_{27}}'\lambda^{-1}\right]\right]\right] \end{aligned} \tag{6.45}$$

$$H_k' = \frac{(6k+1)(6k-7)}{72k}, \qquad k = 1, 2, \ldots, L_{27} \tag{6.46}$$

$$L_{27} = \mathrm{int}\left(4 + 300\lambda^{-1}\right) \tag{6.47}$$

Appendix B. FORTRAN Code Listing

Listed below is the FORTRAN code listing of all the routines. Related routines are grouped together; the single precision, real argument routines appear first, followed by the single precision, complex argument routines, and then the double precision real and complex versions.

Single Precision, Real Argument Routines

RBESSEL	Bessel function driver, user-called
RHANKEL	Hankel function driver, user-called
RBSJ1	Bessel function $J_m(x)$ via power series
RDBSJ1	Bessel function $J_m'(x)$ via power series
RBSY1	Bessel function $Y_m(x)$ via power series
RDBSY1	Bessel function $Y_m'(x)$ via power series
RBESJY2	all Bessel functions via Airy function approximation
RBESJY3	all Bessel functions via asymptotic expansion

Single Precision, Complex Argument Routines

CBESSEL	Bessel function driver, user-called
CHANKEL	Hankel function driver, user-called
CBSJ1	Bessel function $J_m(x)$ via power series
CDBSJ1	Bessel function $J_m'(x)$ via power series
CBSY1	Bessel function $Y_m(x)$ via power series
CDBSY1	Bessel function $Y_m'(x)$ via power series
CBESJY2	all Bessel functions via Airy function approximation
CBESJY3	all Bessel functions via asymptotic expansion

Double Precision, Real Argument Routines

RDBESSEL	Bessel function driver, user-called
RDHANKEL	Hankel function driver, user-called
DRBSJ1	Bessel function $J_m(x)$ via power series
RDFACT	utility: (x/2)^n/n!
DRDBSJ1	Bessel function $J_m'(x)$ via power series
DRBSY1	Bessel function $Y_m(x)$ via power series
DDIGAM	utility: digamma function
DRDBSY1	Bessel function $Y_m'(x)$ via power series
DRBESJY2	all Bessel functions via Airy function approximation
RDAIRYFN	utility: Airy functions
DRBESJY3	all Bessel functions via asymptotic expansion
RDTOLCH	utility: check for under-overflow

Double Precision, Complex Argument Routines

CDBESSEL	Bessel function driver, user-called
CDHANKEL	Hankel function driver, user-called
DCBSJ1	Bessel function $J_m(x)$ via power series
CDFACT	utility: (x/2)^n/n!
DCDBSJ1	Bessel function $J_m'(x)$ via power series
DCBSY1	Bessel function $Y_m(x)$ via power series
DDIGAM	utility: digamma function
DCDBSY1	Bessel function $Y_m'(x)$ via power series
DCBESJY2	all Bessel functions via Airy function approximation
CDAIRYFN	utility: Airy functions
DCBESJY3	all Bessel functions via asymptotic expansion
CDTOLCH	utility: check for under-overflow

```
c*******************************************
      subroutine rbessel(arg,m,icode,besj,besy,dbesj,dbesy)

c     This set of routines calculates various types of Bessel functions
c     in single precision.

c     INPUTS:

c     arg   -- real argument x >= 0
c     m     -- integer order m >= 0
c     icode --  1  calculate   Jm(x), the Bessel fcn of first kind
c               2              Ym(x), the Bessel fcn of second kind
c               3              Jm(x), Ym(x) and their first derivatives

c     OUTPUT:

c     the value of the indicated Bessel function, written to the following
c     variables, according to the value of icode:
c     icode =   1 -->  besj
c               2      besy
c               3      besj,besy,dbesj,dbesy

c     NOTE:  Ym(0) and  Y'm(0) are undefined.

c     Three different evaluation methods are used, depending on the
c     relationship between M and ARG:
c     1) if ARG is small, the power series definition is used directly
c        (routines RBSJ1, RDBSJ1, RBSY1, RDBSY1).
c     2) if both are large, a series involving Airy functions is used
c        (routine RBESJY3).
c     3) if M is small but ARG is large, an asymptotic expansion is used
c        (routine RBESJY2).
```

```
      integer m,icode
      real arg,zarg,zm,besj,dbesj,besy,dbesy
      double precision ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm

      ertolp = 1.0e35
      ertolm = 1.0e-35

      zarg = min( .007*m**2+.16*m+12.5, 40. )
      zm = min( 3.*arg/11.+20., 33. )

      if (arg .le. zarg) then
c        use power series
         if (icode .eq. 1) call rbsj1(arg,m,besj)
         if (icode .eq. 2) call rbsy1(arg,m,besy)
         if (icode .eq. 3) then
            call rbsj1(arg,m,besj)
            call rdbsj1(arg,m,dbesj)
            call rbsy1(arg,m,besy)
            call rdbsy1(arg,m,dbesy)
         endif
      else if (m .ge. zm) then
c        use airy function approximation
         call rbesjy2(arg,m,icode,besj,besy,dbesj,dbesy)
      else
c        use asymptotic expansion approximation
         call rbesjy3(arg,m,icode,besj,besy,dbesj,dbesy)
      endif

      return
      end

c*******************************************
      subroutine rhankel(arg,order,icode,han1,han2,dhan1,dhan2)

c     This set of routines calculates various types of Hankel functions
c     in single precision.

c     INPUTS:

c     arg   -- real argument x >= 0
c     order -- integer order m >= 0
c     icode --    1    calculate  H[1]m(x), the Hankel fcn of the first kind
c                 2               H[2]m(x), the Hankel fcn of the second kind
c                 3               H[1]m(x), H[2]m(x) & first derivatives

c     OUTPUT:

c     the value of the indicated Hankel function, written to the following
```

```
c     variables, according to the value of icode:
c     icode =    1 --> han1
c                2     han2
c                3     han1,han2,dhan1,dhan2

c     NOTE: the variables han1,han2,dhan1,dhan2 must be declared complex
c         in the calling program.

      integer order,icode
      complex han1,dhan1,han2,dhan2,ic
      real arg,besj,dbesj,besy,dbesy

      ic = (0.,1.)

      if (arg .ge. 0.) then
        call rbessel(arg,order,3,besj,besy,dbesj,dbesy)
        if (icode .eq. 1) then
          han1 = besj + ic*besy
        else if (icode .eq. 2) then
          han2 = besj - ic*besy
        else
          han1 = besj + ic*besy
          han2 = besj - ic*besy
          dhan1 = dbesj + ic*dbesy
          dhan2 = dbesj - ic*dbesy
        endif
      else
        arg = -arg
c       Abr-Stegun 9.1.39
        call rbessel(arg,order,3,besj,besy,dbesj,dbesy)
        if (icode .eq. 1) then
          han1 = -(-1.)**order*(besj-ic*besy)
        else if (icode .eq. 2) then
          han2 = -(-1.)**order*(besj+ic*besy)
        else
          han1 = -(-1.)**order*(besj-ic*besy)
          han2 = -(-1.)**order*(besj+ic*besy)
          dhan1 = (-1.)**order*(dbesj-ic*dbesy)
          dhan2 = (-1.)**order*(dbesj+ic*dbesy)
        endif
      endif

      return
      end

c*******************************************
      subroutine rbsj1(arg,m,besj1)

      integer m
      double precision darg,besj,ertolp,ertolm
```

```
      real arg,besj1

      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call drbsj1(darg,m,besj)
      besj1 = besj

      return
      end

c*****************************************
      subroutine rdbsj1(arg,m,dbesj1)

      integer m
      double precision darg,dbesj,ertolp,ertolm
      real arg,dbesj1

      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call drdbsj1(darg,m,dbesj)
      dbesj1 = dbesj

      return
      end

c*****************************************
      subroutine rbsy1(arg,m,besy1)

      integer m
      double precision darg,besy,ertolp,ertolm
      real arg,besy1

      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call drbsy1(darg,m,besy)
      besy1 = besy

      return
      end

c*****************************************
      subroutine rdbsy1(arg,m,dbesy1)

      integer m
      double precision darg,dbesy,ertolp,ertolm
      real arg,dbesy1
```

```
      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call drdbsy1(darg,m,dbesy)
      dbesy1 = dbesy

      return
      end

c******************************************
      subroutine rbesjy2(arg,m,icode,besj2,besy2,dbesj2,dbesy2)

c  Airy function approximation to Bessel function, real argument

      integer m,icode
      double precision darg,besj,besy,dbesj,dbesy,ertolp,ertolm
      real arg,besj2,besy2,dbesj2,dbesy2

      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call drbesjy2(darg,m,icode,besj,besy,dbesj,dbesy)
      besj2 = besj
      besy2 = besy
      dbesj2 = dbesj
      dbesy2 = dbesy

      return
      end

c******************************************
      subroutine rbesjy3(arg,m,icode,besj3,besy3,dbesj3,dbesy3)

c  asymptotic approximation to Bessel function, real argument

      integer m,icode
      double precision darg,besj,besy,dbesj,dbesy,ertolp,ertolm
      real arg,besj3,besy3,dbesj3,dbesy3

      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call drbesjy3(darg,m,icode,besj,besy,dbesj,dbesy)
      besj3 = besj
      besy3 = besy
      dbesj3 = dbesj
      dbesy3 = dbesy

      return
      end
```

```
c******************************************
      subroutine cbessel(arg,m,icode,besjc,besyc,dbesjc,dbesyc)

c     This set of routines calculates various types of Bessel functions
c     in single precision.

c     INPUTS:

c     arg   -- complex argument
c     m     -- integer order m >= 0
c     icode --   1 calculate    Jm(x), the Bessel fcn of first kind
c                2              Ym(x), the Bessel fcn of second kind
c                3              Jm(x),Ym(x) & their first derivatives

c     OUTPUT:

c     the value of the indicated Bessel function, written to the following
c     variables, according to the value of icode:
c     icode =    1 --> besjc
c                2     besyc
c                3     besjc,besyc,dbesjc,dbesyc

c     NOTE:  Ym(0) and Y'm(0) are undefined.

c     Three different evaluation methods are used, depending on the
c     relationship between M and ARG:
c     1) if both are small, the power series definition is used directly
c        (routines CBSJ1, CDBSJ1, CBSY1, CDBSY1).
c     2) if both are large, a series involving Airy functions is used
c        (routine CBESJY3).
c     3) if M is small but ARG is large, an asymptotic expansion is used
c        (routine CBESJY2).

      integer m,icode
      complex arg,besjc,besyc,dbesjc,dbesyc
      real zm
      double precision ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm

      ertolp = 1.0d35
      ertolm = 1.0d-35

      if (m .lt. 8) then
        zm = 7. + 2.*float(m)/7.
      else
        zm = (184. - 8.*float(m))/15.
      endif

      if (abs(arg) .le. zm) then
```

```
c           use power series
      if (icode .eq. 1) call cbsj1(arg,m,besjc)
      if (icode .eq. 2) call cbsy1(arg,m,besyc)
      if (icode .eq. 3) then
            call cbsj1(arg,m,besjc)
            call cdbsj1(arg,m,dbesjc)
            call cbsy1(arg,m,besyc)
            call cdbsy1(arg,m,dbesyc)
      endif
      else if (m .ge. 8) then
c        use airy approximation
         call cbesjy2(arg,m,icode,besjc,besyc,dbesjc,dbesyc)
         else
c        use asymptotic approximation
         call cbesjy3(arg,m,icode,besjc,besyc,dbesjc,dbesyc)
      endif

      return
      end

c******************************************
      subroutine chankel(arg,order,icode,han1c,han2c,dhan1c,dhan2c)

c     This set of routines calculates various types of Hankel functions
c     in single precision.

c     INPUTS:

c     arg   --   complex argument
c     order --   integer order m >= 0
c     icode --   1      calculate H[1]m(x), the Hankel fcn of first kind
c                2      H[2]m(x), the Hankel fcn of second kind
c                3      H[1]m(x),H[2]m(x) & first derivatives

c     OUTPUT:

c     the value of the indicated Hankel function, written to the following
c     variables, according to the value of icode:
c     icode =    1 --> han1c
c                2      han2c
c                3      han1c,han2c,dhan1c,dhan2c


      integer order,icode
      complex arg,han1c,dhan1c,han2c,dhan2c
      complex besjc,dbesjc,besyc,dbesyc,ic

      ic = (0.,1.)
      call cbessel(arg,order,3,besjc,besyc,dbesjc,dbesyc)
```

```
      if (icode .eq. 1) then
        han1c = besjc + ic*besyc
      else if (icode .eq. 2) then
        han2c = besjc - ic*besyc
      else
        han1c = besjc + ic*besyc
        han2c = besjc - ic*besyc
        dhan1c = dbesjc + ic*dbesyc
        dhan2c = dbesjc - ic*dbesyc
      endif

      return
      end

c*****************************************
      subroutine cbsj1(arg,m,besj1)

      integer m
      complex arg,besj1
      double complex darg,besj
      double precision ertolp,ertolm

      common /DERRTOL/  ertolp,ertolm

      darg = arg
      call dcbsj1(darg,m,besj)
      besj1 = besj

      return
      end

c*****************************************
      subroutine cdbsj1(arg,m,dbesj1)

      integer m
      complex arg,dbesj1
      double complex darg,dbesj
      double precision ertolp,ertolm

      common /DERRTOL/  ertolp,ertolm

      darg = arg
      call dcdbsj1(darg,m,dbesj)
      dbesj1 = dbesj

      return
      end
```

```
c*******************************************
      subroutine cbsy1(arg,m,besy1)

      integer m
      complex arg,besy1
      double complex darg,besy
      double precision ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call dcbsy1(darg,m,besy)
      besy1 = besy

      return
      end

c*******************************************
      subroutine cdbsy1(arg,m,dbesy1)

      integer m
      complex arg,dbesy1
      double complex darg,dbesy
      double precision ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call dcdbsy1(darg,m,dbesy)
      dbesy1 = dbesy

      return
      end

c*******************************************
      subroutine cbesjy3(arg,m,icode,besj3,besy3,dbesj3,dbesy3)

c  Airy function approximation to Bessel function, complex argument

      integer m,icode
      complex arg,besj3,besy3,dbesj3,dbesy3
      double complex darg,besj,besy,dbesj,dbesy
      double precision ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm

      darg = arg
      call dcbesjy3(darg,m,icode,besj,besy,dbesj,dbesy)
      besj3 = besj
      besy3 = besy
```

```
      dbesj3 = dbesj
      dbesy3 = dbesy

      return
      end

c*******************************************
      subroutine cbesjy2(arg,m,icode,besj2,besy2,dbesj2,dbesy2)

c  asymptotic approximation to Bessel function, complex argument
c     Abr-Stegun 9.3.35, 9.3.36

      integer m,icode
      complex arg,besj2,besy2,dbesj2,dbesy2
      double complex darg,besj,besy,dbesj,dbesy
      double precision ertolp,ertolm

      common /DERRTOL/  ertolp,ertolm

      darg = arg
      call dcbesjy2(darg,m,icode,besj,besy,dbesj,dbesy)
      besj2  = besj
      besy2  = besy
      dbesj2 = dbesj
      dbesy2 = dbesy

      return
      end

c*******************************************
      subroutine rdbessel(arg,m,icode,besj,besy,dbesj,dbesy)

c     This set of routines calculates various types of Bessel functions
c     in double precision.

c     INPUTS:

c     arg   -- real argument x >= 0
c     m     -- integer order m >= 0
c     icode -- 1  calculate  Jm(x), the Bessel fcn of first kind
c              2             Ym(x), the Bessel fcn of second kind
c              3             Jm(x), Ym(x) and their first derivatives

c  OUTPUT:

c    the value of the indicated Bessel function, written to the following
c    variables, according to the value of icode:
c    icode  = 1 --> besj
c             2     besy
c             3     besj,besy,dbesj,dbesy
```

```
c  NOTE:  Ym(0) and Y'm(0) are undefined.

c  Three different evaluation methods are used, depending on the
c  relationship between M and ARG:
c    1) if both are small, the power series definition is used directly
c       (routines DRBSJ1, DRDBSJ1, DRBSY1, DRDBSY1).
c    2) if both are large, a series involving Airy functions is used
c       (routine DRBESJY3).
c    3) if M is small but ARG is large, an asymptotic expansion is used
c       (routine DRBESJY2).

      integer m,icode
      double precision arg,zm,besj,dbesj,besy,dbesy,ertolp,ertolm
      double precision zarg

      common /DERRTOL/  ertolp,ertolm

      ertolp = 1.0d35
      ertolm = 1.0d-35

      zarg = min( .007*m**2+.16*m+12.5, 40. )
      zm = min( 3.*arg/11.+20., 33. )

      if (arg .le. zarg) then
c        use power series
         if (icode .eq. 1) call drbsj1(arg,m,besj)
         if (icode .eq. 2) call drbsy1(arg,m,besy)
         if (icode .eq. 3) then
            call drbsj1(arg,m,besj)
            call drdbsj1(arg,m,dbesj)
            call drbsy1(arg,m,besy)
            call drdbsy1(arg,m,dbesy)
         endif
      else if (m .ge. zm) then
c        use airy function approximation
         call drbesjy2(arg,m,icode,besj,besy,dbesj,dbesy)
      else
c        use asymptotic expansion approximation
         call drbesjy3(arg,m,icode,besj,besy,dbesj,dbesy)
      endif

      return
      end

c******************************************
      subroutine rdhankel(arg,order,icode,han1,han2,dhan1,dhan2)

c  This set of routines calculates various types of Hankel functions
c  in double precision.
```

```
c     INPUTS:

c     arg   -- real argument x >= 0
c     order -- integer order m >= 0
c     icode --  1   calculate    H[1]m(x), the Hankel fcn of first kind
c               2                H[2]m(x), the Hankel fcn of second kind
c               3                H[1]m(x), H[2]m(x) & first derivatives

c     OUTPUT:

c     the value of the indicated Hankel function, written to the following
c     variables, according to the value of icode:
c     icode   = 1 --> han1
c               2     han2
c               3     han1,han2,dhan1,dhan2

c     NOTE: the variables han1,han2,dhan1,dhan2 must be declared complex.

      integer order,icode
      double complex han1,dhan1,han2,dhan2,ic
      double precision arg,besj,dbesj,besy,dbesy

      ic = (0.d0,1.d0)

      if (arg .ge. 0.d0) then
      call rdbessel(arg,order,3,besj,besy,dbesj,dbesy)
      if (icode .eq. 1) then
         han1 = besj + ic*besy
      else if (icode .eq. 2) then
         han2 = besj - ic*besy
      else
         han1 = besj + ic*besy
         han2 = besj - ic*besy
         dhan1 = dbesj + ic*dbesy
         dhan2 = dbesj - ic*dbesy
      endif
      else
         arg = -arg
c     Abr-Stegun 9.1.39
      call rdbessel(arg,order,3,besj,besy,dbesj,dbesy)
      if (icode .eq. 1) then
         han1 = -(-1.d0)**order*(besj-ic*besy)
      else if (icode .eq. 2) then
         han2 = -(-1.d0)**order*(besj+ic*besy)
      else
         han1 = -(-1.d0)**order*(besj-ic*besy)
         han2 = -(-1.d0)**order*(besj+ic*besy)
         dhan1 = (-1.d0)**order*(dbesj-ic*dbesy)
         dhan2 = (-1.d0)**order*(dbesj+ic*dbesy)
      endif
```

```
      endif

      return
      end

c******************************************
      subroutine drbsj1(arg,m,besj1)

      integer m,lone,k,mone
      double precision arg,besj1,fm,qntone,fmone,f,rdfact,ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm

      f = rdfact(m,arg)
      if (f .lt. 1.5d0*ertolm) then
        besj1 = 0.0d0
        return
      endif
      if (f .gt. .5d0*ertolp) then
        besj1 = ertolp
        return
      endif

      fm = dble(m)
      lone = int(10.0d0 + 4.0d0*arg/3.0d0)
      qntone = 1.0d0

      do 10 k = 1,lone
        mone = lone - k + 1
        fmone = dble(mone)
        qntone = 1.0d0 - qntone*(0.5d0*arg)**2/(fmone*(fmone + fm))
10    continue

      besj1 = f*qntone
      return
      end

c******************************************
      function rdfact(n,arg)

c   calculate (arg/2)^n/n!

      integer n
      double precision rdfact,arg,flag,ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm

      rdfact = 1.0d0
      if (n .eq. 0) return
```

```
      do 5 k = 1,n
        rdfact = rdfact*arg/(2.0d0*dble(n-k+1))
        if (rdfact .lt. ertolm) then
          rdfact = ertolm
          return
        endif
        if (rdfact .gt. ertolp) then
          rdfact = ertolp
          return
        endif
5     continue
      return
      end

c******************************************
      subroutine drdbsj1(arg,m,dbesj1)

      integer m,lexit,k,mexit
      double precision arg,dbesj1,fm,dqnton,amexit,f,rdfact
      double precision ertolp,ertolm,besj1

      common /DERRTOL/ ertolp,ertolm

      if (m .eq. 0) then
        call drbsj1(arg,1,besj1)
        dbesj1 = -besj1
        return
      endif

      f = rdfact(m,arg)
      if (f .lt. 1.5d0*ertolm) then
        dbesj1 = 0.0d0
        return
      endif
      if (f .gt. .5d0*ertolp) then
        dbesj1 = ertolp
        return
      endif

      fm = dble(m)
      lexit = int(10.0d0 + 1.6d0*arg)
      dqnton = 1.0d0
      do 10 k = 1,lexit
        mexit = lexit - k + 1
        amexit = dble(mexit)
        dqnton = 1.0d0 - dqnton*(0.5d0*arg)**2*(fm+2.0d0*amexit) /
     .          (amexit*(fm+amexit)*(fm+2.0d0*amexit-2.0d0))
10    continue

      dbesj1 = 0.5d0*dqnton*rdfact(m-1,arg)
```

```
      return
      end

c*******************************************
      subroutine drbsy1(arg,m,besy1)

      integer m,k,lz,klz,iflag
      double precision arg,besy1,fm,gamma,pi,repi,emz,besj1,pemzj,qntbz
      double precision am,sumdk,gmz
      double precision rdfact,f,fmz,ertolp,ertolm,qntdk,fklz,dk,ddigam

      common /DERRTOL/ ertolp,ertolm

      f = rdfact(m,arg)
      fm = dble(m)
      gamma = 0.57721566490153d0
      pi = 3.1415926535897932d0
      repi = 1.0d0/pi

      call rdtolch(arg,besy1,0.0d0,-ertolp,iflag)
      if (iflag .eq. 1) return

      emz = 2.0d0*repi*(gamma + dlog(0.5d0*arg))
      call drbsj1(arg,m,besj1)
      pemzj = emz*besj1

      if (m .eq. 0) then
        fmz = 0.0d0
      else if (m .eq. 1) then
        fmz = -2.d0*repi/arg
      else if (f .lt. 1.5d0*ertolm) then
        fmz = -ertolp
      else if (f .gt. .5d0*ertolp) then
        fmz = 0.0d0
      else
        qntbz = 1.0d0
        do 10 k = 1,m-1
          am = dble(m-k)
          qntbz = 1.0d0 + qntbz*(0.5d0*arg)**2/(am*(fm-am))
10      continue
        fmz = -qntbz*repi / (f*fm)
      endif

      lz = int(7.0d0+1.5d0*arg)
      qntdk = 1.0d0
      do 20 k = 1,lz
        klz = lz - k + 1
        fklz = dble(klz)
        dk = (ddigam(klz+1) + ddigam(m+klz+1)) /
     .      ((ddigam(klz)+ddigam(klz+m))*(fklz+1.)*(fm+fklz+1.))
```

```
      qntdk = 1.0d0 - qntdk*dk*(0.5d0*arg)**2
20   continue
     sumdk = ddigam(m)-(qntdk*(1.d0+ddigam(m+1))*(arg*0.5d0)**2 /
    .        (fm+1.0d0))

     if (f .lt. 1.5d0*ertolm) then
       gmz = 0.0d0
     else if (f .gt. .5d0*ertolp) then
       gmz = ertolp
     else
       gmz = -sumdk*repi*f
     endif

     besy1 = pemzj + fmz + gmz
     return
     end

c******************************************
     function ddigam(n)

     integer n,j
     double precision ddigam

     ddigam = 0.0d0
     if (n .eq. 0) return
     do 5 j = n,1,-1
       ddigam = ddigam + 1.d0/dble(j)
5    continue

     return
     end

c******************************************
     subroutine drdbsy1(arg,m,dbesy1)

     integer m,k,mm,kmm,larg,klarg,iflag
     double precision arg,dbesy1,fm,gamma,pi,repi,emz,dbesj1,demzj1
     double precision demzj2,qntdb,f,fk,dfmz,qntder,flarg,dkder,ddigam
     double precision dergmz,ertolp,ertolm,rdfact,besj1,sumder

     common /DERRTOL/ ertolp,ertolm

     f = rdfact(m,arg)
     fm = dble(m)
     gamma = 0.57721566490153d0
     pi = 3.1415926535897932d0
     repi = 1.0d0/pi

     call rdtolch(arg,dbesy1,0.0d0,ertolp,iflag)
     if (iflag .eq. 1) return
```

```
      emz = 2.0d0*repi*(gamma + dlog(0.5d0*arg))
      call drdbsj1(arg,m,dbesj1)
      demzj1 = emz*dbesj1
      call drbsj1(arg,m,besj1)
      demzj2 = besj1*2.0d0*repi/arg

      if (m .eq. 0) then
        dfmz = 0.0d0
      else if (m .eq. 1) then
        dfmz = 2.0d0*repi/arg**2
      else
        qntdb = 1./(.5d0*arg*f)
        k = 1
        mm = m - 1
10      kmm = mm - k
        fk = dble(k)
        qntdb = qntdb + ((fm-(2.d0*fk))*rdfact(k,arg)) /
     .              (rdfact(kmm,arg)*(.5d0*arg)**2)
        if (abs(qntdb) .lt. 1.5d0*ertolm) then
          dfmz = 0.0d0
        else if (qntdb .gt. .5d0*ertolp) then
          dfmz = ertolp
        else
          dfmz = .5d0*qntdb*repi
        endif
        k = k + 1
        if (k .le. mm) go to 10
      endif

      if (f .lt. 1.5d0*ertolm) then
        dergmz = 0.0d0
      else if (f .gt. .5d0*ertolp) then
        dergmz = ertolp
      else
        larg = int(8.0d0+1.4d0*arg)
        qntder = 1.0d0
        do 20 k = 1,larg
          klarg = larg - k + 1
          flarg = dble(larg-k+1)
          dkder = (ddigam(klarg+1) + ddigam(m+klarg+1)) *
     .            (fm+2.d0*flarg+2.d0)
     .         / ( (ddigam(klarg)+ddigam(m+klarg))*(flarg +1.d0)*
     .             (fm+flarg +1.d0)*(fm+2.d0*flarg) )
          qntder = 1.0d0 - qntder*dkder*(0.5d0*arg)**2
20      continue
        sumder = fm*ddigam(m) - qntder*(1.d0+ddigam(m+1))*
     .          (fm + 2.d0)*(0.5d0*arg)**2/(fm+1.d0)
        dergmz = -0.5d0*repi*f*sumder/(0.5d0*arg)
      endif
```

```
      dbesy1 = demzj1 + demzj2 + dfinz + dergmz
      call rdtolch(dbesy1,dbesy1,ertolp,0.0d0,iflag)
      return
      end

c*******************************************
      subroutine drbesjy2(arg,m,icode,besj2,besy2,dbesj2,dbesy2)

c   Airy function approximation to Bessel function, real argument

      integer m,icode
      double precision arg,besj2,besy2,dbesj2,dbesy2,fm,x,zeta,q1,phi
      double precision q1i,g3,g32,g2,gsq,a1,b0,c0,d1,h,y,airy,biry
      double precision ertolp,ertolm,f1,f2,f3,dairy,dbiry
      double precision u1,u2,u3,b1,q,u4,u5,a2,b2

      common /DERRTOL/ ertolp,ertolm

      fm = dble(m)
      x = arg/fm
      if (x .gt. 1.0d0) then
        zeta = -(1.5d0*(sqrt(x**2-1.d0) - acos(1.d0/x)))**(2.d0/3.d0)
      else
        zeta = (1.5d0*(dlog((1.0d0 +sqrt(1.d0-x**2))/x)
     .        - sqrt(1.d0-x**2)))**(2.d0/3.d0)
      endif

      if (x .gt. 0.98d0 .and. x .lt. 1.02d0) then
        h = 1.0d0 - x
        phi = 2.0d0**(1.d0/3.d0)*(1.0d0 + 0.2d0*h + 3.d0*h**2/35.d0
     .      + 73.d0*h**3/1575.d0 + 35209.d0*h**4/1212750.d0 +
     .        380069.d0*h**5/18768750.d0)
        a1 = -1.d0/225.d0 - 71.d0*h/38500.d0
        b0 = (1.d0/70.d0 + 2.d0*h/225.d0)*2.d0**(1.d0/3.d0)
        b1 = 2.d0**(1.d0/3.)*(-1213.d0/1023750.d0 - 3757.d0*h/2695000.
     .     - h**2*(8.9962899979797d-4 + h*(.0002753433716d0 - h*
     .     (.00018048868d0 + h*.0004108523))))
        a2 = 6.937355413546877d-4 + h*(.00046448349036601 - h*
     .     (.0002890362546053d0 + h*(.0008747649439535d0 +
     .     h*.00102971637614)))
        b2 = -2.d0**(1./3.)*(4.382918094497229d-4 + h*
     .     (7.110486511691 1d-4 + h*5.318984348085d-4))
        b2 = b2 - 2.d0**(1./3.)*h**3*2.182958472d-4
        c0 = (0.1d0 + 0.02d0*h)*2.d0**(2.d0/3.d0)
        d1 = 23.d0/3150.d0 + 1453.d0*h/346500.d0

      else
        q1 = zeta/(1.0d0-x**2)
        phi = (4.0d0*q1)**0.25d0
        f1 = 1.0d0 - x**2
```

```
      f2 = f1**2
      f3 = f1**3
      q1i = 1.0d0/q1
      g3 = q1i**3
      g32 = q1i**1.5d0
      g2 = q1i**2
      gsq = sqrt (q1i)
      a1 = ( -455.d0*g3/4608.d0-7.d0*g32*(f1-5.d0/3.d0)/384.d0 +
     .    385.d0/1152.d0 - 77.d0*f1/192.d0 + 9.d0*f2/128.d0 ) / f3
      b0 = (-5.d0*g2/48.d0 + gsq*(5.d0/24.d0 - f1/8.d0))/f2
      c0 = (7.d0*q1i/48.d0 - 3.d0*sqrt(q1)*(7.d0/72.d0 - f1/8.d0))/f1
      d1 = ( 385.d0*g3/4608.d0 + 5.d0*g32*(-f1 + 7.d0/9.d0)/128.d0 -
     .    15.d0*f2/128.d0 + 33.d0*f1/64.d0 - 455.d0/1152.d0 ) / f3

      q = sqrt(f1/zeta)
      u1 = q*(1.d0-5.d0/(f1*3.d0)) / (8.d0*f1)
      u2 = (4.5d0 - 77.d0/(f1*3.d0)*(1.d0-5.d0/(f1*6.d0))) /
     .    (64.d0*f1)
      u3 = q*(75.d0/2. - 456.3d0/f1 + 17017.d0/(f2*18.d0)*(1.d0-
     .        5.d0/(f1*9.d0))) / (512.d0*f2)
      u4 = (3675.d0/8.d0 - 9683.3d0/f1 + 2717.d0/f2*(53.d0/4.d0 -
     .    2737.d0/(f1*162.d0)*(1.d0-5.d0/(12.d0*f1))))/(4096.d0*f2)
      u5 = 59535.d0/8.d0 - 221.d0/(4.d0*f1)*(305923.d0/70.d0 -
     .    77.d0/f1*(14743.d0/45.d0 - 95.d0/f1*(67.d0/9.d0-
     .    3335.d0/(486.d0*f1)*(1.d0-1.d0/(3.d0*f1)))))
      u5 = q*u5/(f3*32768.d0)
      b1 = -u3 - 5.d0/(zeta**2*48.d0)*(u2+77.d0/(zeta*96.d0)*
     .    (u1 + 221.d0/(zeta**2*144.d0)))
      b2 = -u5 - 5.d0/(zeta**2*48.d0)*(u4 + 77.d0/(zeta*96.d0)*
     .    (u3 + 221.d0/(zeta**2*144.d0)*(u2 + 437.d0/(zeta*192.d0)*
     .    (u1 + 145.d0/(zeta**2*48.d0)))))
      a2 = u4 - 7.d0/(zeta*48.d0)*(u3 + 65.d0/(zeta**2*96.d0)*(u2 +
     .    209.d0/(zeta*144.d0)*(u1 + 425.d0/(zeta**2*192.d0))))
    endif

    y = zeta*fm**(2.d0/3.d0)
    call rdairyfn(y,icode,airy,biry,dairy,dbiry)

    if (icode .eq. 1) then
      besj2 = phi/(fm**(1.d0/3.d0))*(airy*(1.d0+a1/fm**2+a2/fm**4)
     .       + dairy / (fm**(4.d0/3.d0))*(b0+b1/fm**2+b2/fm**4))
    else if (icode .eq. 2) then
      if (biry .eq. ertolp .or. dbiry .eq. ertolp) then
        besy2 = -ertolp
      else
        besy2 = -phi/(fm**(1./3.))*(biry*(1.d0+a1/fm**2+a2/fm**4)
     .          + dbiry / (fm**(4.d0/3.d0))*(b0+b1/fm**2+b2/fm**4))
      endif
    else
```

```
      besj2 = phi/(fm**(1.d0/3.d0))*(airy*(1.d0+a1/fm**2+a2/fm**4)
     .        + dairy / (fm**(4.d0/3.d0))*(b0+b1/fm**2+b2/fm**4))

      if (biry .eq. ertolp .or. dbiry .eq. ertolp) then
        besy2 = -ertolp
      else
        besy2 = -phi/(fm**(1./3.))*(biry*(1.d0+a1/fm**2+a2/fm**4)
     .        + dbiry / (fm**(4.d0/3.d0))*(b0+b1/fm**2+b2/fm**4))
      endif

      dbesj2 = -2.d0/( fm**(2.d0/3.d0)*x*phi)*(airy/(fm**(2.d0/3.d0))
     .        *c0 + dairy*(1.d0 + d1/(fm**2.d0)) )

      if (biry .eq. ertolp .or. dbiry .eq. ertolp) then
        dbesy2 = ertolp
      else
        dbesy2 = 2.d0/(fm**(2.d0/3.d0)*x*phi)*(biry*c0/
     .        (fm**(2.d0/3.d0)) + dbiry*(1.d0 + d1/(fm**2.d0)))
      endif
      endif

      return
      end

c*****************************************
      subroutine rdairyfn(y,icode,airy,biry,dairy,dbiry)

      integer icode,l1,k,k2,l3,l4,l5,m5,lp,lpt
      double precision y,airy,biry,dairy,dbiry,c1,c2,flam,reflam,pi
      double precision clam,srtwo,qntv,ffm1,vffm1,vffmr,qntw,ffm2,wffmr
      double precision qntdv,ffm3,dvffmr,qntdw,ffm4,dwffmr,sum1,sum2
      double precision sum4,dift1,sumt1,dift2,sumt2,qntf,qntg,qntdf
      double precision ff,fg,fdf,fdg,fml,fpl,fmp,gml,gpl,fmpt,sxfmpt
      double precision ertolp,ertolm,srpi,aby,slam,sum3,ffm5,qntdg
      double precision sxfmp

      common /DERRTOL/ ertolp,ertolm

      c1 = 0.355028053887817d0
      c2 = 0.258819403792807d0

      if (y .gt. -10.0d0 .and. y .lt. 6.0d0) then

c       power series
        if (y .ge. 0.0d0) then
          L5 = int(5.0d0 + 3.0d0*dabs(y))
        else
          L5 = int(5.0d0 + 3.5d0*dabs(y))
        endif
        qntf = 1.0d0
```

```
        qntg = 1.0d0
        qntdf = 1.0d0
        qntdg = 1.0d0
        do 84 k = 1,L5
          m5 = L5 - k + 1
          ffm5 = 3.d0*dble(m5)
          qntf = 1.0d0 + qntf*y**3/(ffm5*(ffm5 - 1.0d0))
          qntg = 1.0d0 + qntg*y**3/(ffm5*(ffm5 + 1.0d0))
          qntdf = 1.0d0 + qntdf*y**3/(ffm5*(ffm5 + 2.0d0))
          qntdg = 1.0d0 + qntdg*y**3/(ffm5*(ffm5-2.0d0))
84       continue
        ff = qntf
        fg = qntg*y
        fdf = qntdf*0.5d0*y**2
        fdg = qntdg
        airy = c1*ff - c2*fg
        dairy = c1*fdf - c2*fdg
        biry = dsqrt(3.0d0)*(c1*ff + c2*fg)
        dbiry = dsqrt(3.0d0)*(c1*fdf + c2*fdg)

      else

        flam = (2.d0/3.d0)*dabs(y)**1.5d0
        reflam = 1.0d0/flam
        pi = 3.1415926535897932d0
        srpi = dsqrt(pi)
        aby = dabs(y)**0.25d0

        if (y .gt. 25.d0) then
          airy = 0.0d0
          dairy = 0.0d0
          biry = ertolp
          dbiry = ertolp
        else if (y .ge. 6.0d0 .and. y .le. 25.0d0) then

c         exponential approximation
          lp = int(4.d0 + 300.0d0*reflam)
          if (reflam .gt. .06) lp = 22 + 11*min((reflam-.06)/.003,
     .                                (.102-reflam)/.039)
          fml = 1.0d0
          fpl = 1.0d0
          do 115 k = 1,lp
            sxfmp = 6.0d0*dble(lp-k+1)
            fml = 1.d0 - fml*reflam*(sxfmp-5.d0)*(sxfmp-1.d0)
     .           /(12.d0*sxfmp)
            fpl = 1.d0 + fpl*reflam*(sxfmp-5.d0)*(sxfmp-1.d0)
     .           /(12.d0*sxfmp)
115        continue
          lpt = lp
          gml = 1.0d0
```

```
         gpl = 1.0d0
         do 125 k = 1,lpt
           sxfmpt = 6.0d0*dble(lpt-k+1)
           gml = 1.d0-gml*reflam*(sxfmpt+1.d0)*(sxfmpt-7.d0)
     .          /(22.d0*sxfmpt)
           gpl = 1.d0+gpl*reflam*(sxfmpt+1.d0)*(sxfmpt-7.d0)
     .          /(12.d0*sxfmpt)
 125     continue
         airy = 0.5d0*y**(-0.25d0)*fml*exp(-flam)/srpi
         dairy = -0.5d0*y**0.25d0*gml*exp(-flam)/srpi
         biry = y**(-0.25d0)*fpl*exp(flam)/srpi
         dbiry = y**0.25d0*gpl*exp(flam)/srpi

       else

c        trigonometric approximation
         slam = sin(flam)
         clam = cos(flam)
         srtwo = sqrt(2.d0)/2.d0
         L1  = int(2.0d0 + 160.0d0*reflam)
         qntv = 1.0d0
         do 20 k = 1,L1
           ffm1 = 12.d0*dble(L1-k+1)
           vffmr = (ffm1-11.d0)*(ffm1-7.d0)*(ffm1-5.d0)*(ffm1-1.d0)
     .           / (144.d0*ffm1*(ffm1 - 6.d0))
           qntv = 1.0d0 - (qntv*vffmr*reflam**2.d0)
 20      continue
         L2  = int(2.0d0 + 220.0d0*reflam)
         qntw = 1.0d0
         do 30 k = 1,L2
           ffm2 = 12.d0*dble(L2-k+1)
           wffmr = (ffm2-5.d0)*(ffm2-1.d0)*(ffm2+1.d0)*(ffm2+5.d0)
     .           / (144.d0*ffm2*(ffm2 + 6.d0))
           qntw = 1.0d0 - (qntw*wffmr*reflam**2.d0)
 30      continue
         L3  = int(2.0d0 + 130.0d0*reflam)
         qntdv = 1.0d0
         do 40 k = 1,L3
           ffm3 = 12.d0*dble(L3-k+1)
           dvffmr = (ffm3-7.d0)*(ffm3-1.d0)*(ffm3+1.d0)*(ffm3+7.d0)
     .           / (144.d0*ffm3*(ffm3 + 6.d0))
           qntdv = 1.0d0 - (qntdv*dvffmr*reflam**2.d0)
 40      continue
         L4  = int(2.0d0 + 200.0d0*reflam)
         qntdw = 1.0d0
         do 50 k = 1,L4
           ffm4 = 12.d0*dble(L4-k+1)
           dwffmr = (ffm4-13.d0)*(ffm4-7.d0)*(ffm4-5.d0)*(ffm4+1.d0)
     .           / (144.d0*ffm4*(ffm4 - 6.d0))
           qntdw = 1.0d0 - (qntdw*dwffmr*reflam**2.d0)
```

```
50          continue
        sum1 = qntv/srpi
        sum2 = 5.d0*qntw*reflam/(72.d0*srpi)
        sum3 = -72.d0*qntdv*reflam/(7.d0*srpi)
        sum4 = qntdw/srpi
        dift1 = srtwo*(sum1 - sum2)
        sumt1 = srtwo*(sum1 + sum2)
        dift2 = srtwo*(sum4 - sum3)
        sumt2 = srtwo*(sum4+sum3)

        airy = (dift1*clam + sumt1*slam)/aby
        dairy = (dift2*slam - sumt2*clam)*aby
        biry = (sumt1*clam - dift1*slam)/aby
        dbiry = (sumt2*slam + dift2*clam)*aby

      endif

    endif

    return
    end

c*******************************************
    subroutine drbesjy3(arg,m,icode,besj3,besy3,dbesj3,dbesy3)

c  asymptotic approximation to Bessel function, real argument

    integer m,icode,lmk,k
    double precision arg,besj3,besy3,dbesj3,dbesy3,fm,pi,srarg,angle
    double precision qnta,qntb,flmk4,amk,bmk,umk,vmk,qntad,qntbd,amkd
    double precision smk,ertolp,ertolm,fmsr,bmkd,rmk

    common /DERRTOL/  ertolp,ertolm

    fm = dble(m)
    pi = 3.1415926535897932d0
    srarg = sqrt(2.0d0/(pi*arg))
    angle = arg - 0.25d0*pi*(1.0d0 + 2.0d0*fm)
    fmsr = 4.0d0*fm**2
    lmk = int( 2 + m/3. + 27./sqrt(arg-12.) )
    if (lmk .lt. 1) lmk = 1

    qnta = 1.0d0
    qntb = 1.0d0
    do 10 k = 1,lmk
      flmk4 = 4.0d0*dble(lmk-k+1)
      amk = (4.0d0*(fmsr-(flmk4-3.0d0)**2)*(fmsr-(flmk4-1.0d0)**2)) /
   .        (flmk4*(flmk4 - 2.0d0))
      bmk = (4.0d0*(fmsr-(flmk4-1.0d0)**2)*(fmsr-(flmk4+1.0d0)**2)) /
   .        (flmk4*(flmk4 + 2.0d0))
```

```
         qnta = 1.0d0 - qnta*amk/(8.0d0*arg)**2
         qntb = 1.0d0 - qntb*bmk/(8.0d0*arg)**2
 10   continue
      umk = qnta
      vmk = qntb*(fmsr - 1.0d0)/(8.0d0*arg)
      besj3 = srarg*(umk*cos(angle) - vmk*sin(angle))
      besy3 = srarg*(umk*sin(angle) + vmk*cos(angle))
      call rdtolch(besj3,besj3,ertolp,0.0d0,iflag)
      call rdtolch(besy3,besy3,ertolp,0.0d0,iflag)

      if (icode .eq. 3) then
         qntad = 1.0d0
         qntbd = 1.0d0
         do 20 k = 1,lmk
            flmk4 = 4.0d0*dble(lmk-k+1)
            amkd = (4.0d0*(fmsr+(flmk4**2-1.0d0))*
     .             (fmsr-(flmk4-3.0d0)**2)*(fmsr-(flmk4-5.0d0)**2)) /
     .             (flmk4*(flmk4 - 2.0d0)*(fmsr+(flmk4-4.0d0)**2-1.0d0))
            bmkd = (4.0d0*(fmsr+(flmk4+2.0d0)**2-1.0d0)*
     .             (fmsr-(flmk4-3.0d0)**2)*(fmsr-(flmk4-1.0d0)**2)) /
     .             (flmk4*(flmk4+2.0d0)*(fmsr+(flmk4-2.0d0)**2-1.0d0))
            qntad = 1.0d0 - qntad*amkd/(8.0d0*arg)**2
            qntbd = 1.0d0 - qntbd*bmkd/(8.0d0*arg)**2
 20      continue
         rmk = qntad
         smk = qntbd*(fmsr + 3.0d0)/(8.0d0*arg)
         dbesj3 = -srarg*(rmk*sin(angle) + smk*cos(angle))
         dbesy3 = srarg*(rmk*cos(angle) - smk*sin(angle))
         call rdtolch(dbesj3,dbesj3,ertolp,0.0d0,iflag)
         call rdtolch(dbesy3,dbesy3,ertolp,0.0d0,iflag)
      endif

      return
      end

c*******************************************
      subroutine rdtolch(rin,rout,setbig,setsmall,iflag)

c   real, double precision check to avoid over- and underflow

      double precision rin,rout,setbig,setsmall,ertolp,ertolm
      integer iflag
      common /DERRTOL/  ertolp,ertolm

      iflag = 0
      if (dabs(rin) .lt. 1.5d0*ertolm) then
         rout = setsmall
         iflag = 1
      endif
      if (dabs(rin) .gt. .5d0*ertolp) then
```

```
      rout = setbig
      iflag = 1
    endif
    return
    end

c****************************************
    subroutine cdbessel(arg,m,icode,besjc,besyc,dbesjc,dbesyc)

c  This set of routines calculates various types of Bessel functions
c  in double precision.

c  INPUTS:

c   arg   -- complex argument
c   m     -- integer order m >= 0
c   icode -- 1   calculate Jm(x), the Bessel fcn of first kind
c            2             Ym(x), the Bessel fcn of second kind
c            3             Jm(x),Ym(x) & their first derivatives

c  OUTPUT:

c   the value of the indicated Bessel function, written to the following
c   variables, according to the value of icode:
c     icode = 1 --> besjc
c             2     besyc
c             3     besjc,besyc,dbesjc,dbesyc

c  NOTE:  Ym(0) and Y'm(0) are undefined.

c  Three different evaluation methods are used, depending on the
c  relationship between M and ARG:
c   1) if both are small, the power series definition is used directly
c      (routines DCBSJ1, DCDBSJ1, DCBSY1, DCDBSY1).
c   2) if both are large, a series involving Airy functions is used
c      (routine DCBESJY3).
c   3) if M is small but ARG is large, an asymptotic expansion is used
c      (routine DCBESJY2).

    integer m,icode
    double complex arg,besjc,besyc,dbesjc,dbesyc
    double precision ertolp,ertolm,zm

    common /DERRTOL/  ertolp,ertolm

    ertolp = 1.0d35
    ertolm = 1.0d-35

    if (m .lt. 8) then
      zm = 7.d0 + 2.d0*dble(m)/7.d0
```

```
      else
        zm = (184.d0 - 8.d0*dble(m))/15.d0
      endif

      if (zabs(arg) .le. zm) then
c        use power series
         if (icode .eq. 1) call dcbsj1(arg,m,besjc)
         if (icode .eq. 2) call dcbsy1(arg,m,besyc)
         if (icode .eq. 3) then
            call dcbsj1(arg,m,besjc)
            call dcdbsj1(arg,m,dbesjc)
            call dcbsy1(arg,m,besyc)
            call dcdbsy1(arg,m,dbesyc)
         endif
      else if (m .ge. 8) then
c        use airy approximation
         call dcbesjy2(arg,m,icode,besjc,besyc,dbesjc,dbesyc)
      else
c        use asymptotic approximation
         call dcbesjy3(arg,m,icode,besjc,besyc,dbesjc,dbesyc)
      endif

      return
      end

c******************************************
      subroutine cdhankel(arg,order,icode,han1c,han2c,dhan1c,dhan2c)

c  This set of routines calculates various types of Hankel functions
c  in double precision.

c  INPUTS:

c    arg   -- complex argument
c    order -- integer order m >= 0
c    icode -- 1 calculate  H[1]m(x), the Hankel fcn of first kind
c             2            H[2]m(x), the Hankel fcn of second kind
c             3            H[1]m(x),H[2]m(x) & first derivatives

c  OUTPUT:

c    the value of the indicated Hankel function, written to the following
c    variables, according to the value of icode:
c    icode = 1 --> han1c
c            2     han2c
c            3     han1c,han2c,dhan1c,dhan2c


      integer order,icode
      double complex arg,han1c,dhan1c,han2c,dhan2c
```

```
      double complex besjc,dbesjc,besyc,dbesyc,ic

      ic = (0.d0,1.d0)

      call cdbessel(arg,order,3,besjc,besyc,dbesjc,dbesyc)
      if (icode .eq. 1) then
        han1c = besjc + ic*besyc
      else if (icode .eq. 2) then
        han2c = besjc - ic*besyc
      else
        han1c = besjc + ic*besyc
        han2c = besjc - ic*besyc
        dhan1c = dbesjc + ic*dbesyc
        dhan2c = dbesjc - ic*dbesyc
      endif

      return
      end

c*****************************************
      subroutine dcbsj1(arg,m,besj1)

      integer m,lone,k
      double complex arg,besj1,qntone,f,cdfact,ic
      double precision ertolp,ertolm,fm,fmone,theta

      common /DERRTOL/ ertolp,ertolm

      f = cdfact(m,arg)

      fm = dble(m)
      ic = (0.d0,1.d0)
      if (zabs(f) .lt. 1.5d0*ertolm) then
        besj1 = 0.0d0
      else if (zabs(f) .gt. .5d0*ertolp) then
        theta = datan(dimag(f)/dble(f))
        besj1 = ertolp*exp(ic*fm*theta)
      else
        lone = int(10.0d0 + 4.0d0*zabs(arg)/3.0d0)
        qntone = 1.0d0
        do 10 k = 1,lone
          fmone = dble(lone - k + 1)
          qntone = 1.0d0 - qntone*(0.5d0*arg)**2/(fmone*(fmone+fm))
10      continue
        besj1 = f*qntone
      endif

      return
      end
```

```
c*******************************************
      function cdfact(n,arg)

c  calculate (arg/2)^n/n!

      integer n,k
      double complex arg,ic,cdfact,f
      double precision theta,ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm

      ic = (0.d0,1.d0)
      f = 1.0d0
      if (n .eq. 0) then
        cdfact = 1.0d0
      else
        if (zabs(arg) .lt. 1.5d0*ertolm) then
          cdfact = ertolm
          return
        endif
        do 5 k = 1,n
          f = f*arg/(2.0d0*dble(n-k+1))
          theta = datan(dimag(f)/dble(f))
          if (zabs(f) .lt. ertolm) then
            cdfact = ertolm*exp(ic*dble(n)*theta)
            return
          endif
          if (zabs(f) .gt. ertolp) then
            cdfact = ertolp*exp(ic*dble(n)*theta)
            return
          endif
5       continue
        cdfact = f
      endif

      return
      end

c*******************************************
      subroutine dcdbsj1(arg,m,dbesj1)

      integer m,lexit,k
      double complex arg,dbesj1,dqnton,f,cdfact,ic,besj1
      double precision fm,amexit,theta,ertolp,ertolm

      common /DERRTOL/ ertolp,ertolm


      ic = (0.d0,1.d0)
      if (m .eq. 0) then
```

```
        call dcbsj1(arg,1,besj1)
        dbesj1 = -besj1
        return
      endif

      f = cdfact(m,arg)
      if (zabs(f) .lt. 1.5d0*ertolm) then
        dbesj1 = 0.0d0
        return
      endif
      if (zabs(f) .gt. .5d0*ertolp) then
        theta = datan(dimag(f)/dble(f))
        dbesj1 = ertolp*exp(ic*dble(m)*theta)
        return
      endif

      fm = dble(m)
      lexit = int(10.0d0 + 1.6d0*zabs(arg))
      dqnton = 1.0d0
      do 10 k = 1,lexit
        amexit = dble(lexit-k+1)
        dqnton = 1.0d0 - dqnton*(0.5d0*arg)**2*(fm+2.0d0*amexit) /
     .          (amexit*(fm+amexit)*(fm+2.0d0*amexit-2.0d0))
10    continue

      dbesj1 = 0.5d0*dqnton*cdfact(m-1,arg)
      return
      end

c******************************************
      subroutine dcbsy1(arg,m,besy1)

      integer m,k,lz,klz,iflag
      double complex arg,besy1,emz,besj1,pemzj,qntbz
      double complex cdfact,f,fmz,qntdk,sumdk,ic,gmz
      double precision fm,gamma,pi,repi,am,ertolp,ertolm,fklz,dk,ddigam
      double precision theta

      common /DERRTOL/ ertolp,ertolm

      ic = (0.d0,1.d0)
      f = cdfact(m,arg)
      fm = dble(m)
      gamma = 0.57721566490153d0
      pi = 3.1415926535897932d0
      repi = 1.0d0/pi

      if (zabs(arg) .lt. 1.5d0*ertolm) then
        besy1 = -ertolp
        return
```

```
      endif

      emz = 2.0d0*repi*(gamma + zlog(0.5d0*arg))
      call dcbsj1(arg,m,besj1)
      pemzj = emz*besj1

      if (m .eq. 0) then
        fmz = 0.0d0
      else if (m .eq. 1) then
        fmz = -2.d0*repi/arg
      else if (zabs(f) .lt. 1.5d0*ertolm) then
        theta = datan(dimag(f)/dble(f))
        fmz = -ertolp*exp(ic*fm*theta)
      else if (zabs(f) .gt. .5d0*ertolp) then
        fmz = 0.0d0
      else
        qntbz = 1.0d0
        do 10 k = 1,m-1
          am = dble(m-k)
          qntbz = 1.0d0 + qntbz*(0.5d0*arg)**2/(am*(fm-am))
10      continue
        fmz = -qntbz*repi / (f*fm)
      endif

      lz = int(7.0d0+1.5d0*zabs(arg))
      qntdk = 1.0d0
      do 20 k = 1,lz
        klz = lz - k + 1
        fklz = dble(klz)
        dk = (ddigam(klz+1) + ddigam(m+klz+1)) /
     .       ((ddigam(klz)+ddigam(klz+m))*(fklz+1.d0)*(fm+fklz+1.d0))
        qntdk = 1.0d0 - qntdk*dk*(0.5d0*arg)**2
20    continue
      sumdk = ddigam(m)-(qntdk*(1.d0+ddigam(m+1))*(arg*0.5d0)**2 /
     .       (fm+1.0d0))

      if (zabs(f) .lt. 1.5d0*ertolm) then
        gmz = 0.0d0
      else if (zabs(f) .gt. .5d0*ertolp) then
        theta = datan(dimag(f)/dble(f))
        gmz = ertolp*exp(ic*fm*theta)
      else
        gmz = -sumdk*repi*f
      endif

      besy1 = pemzj + fmz + gmz
      call cdtolch(besy1,besy1,ertolp,0.0d0,iflag)
      return
      end
```

```
c*******************************************
      subroutine dcdbsy1(arg,m,dbesy1)

      integer m,k,kmm,larg,klarg
      double complex arg,dbesy1,emz,dbesj1,demzj1,besj1
      double complex demzj2,qntdb,f,dfmz,qntder,sumder
      double complex dergmz,cdfact,ic
      double precision fm,gamma,pi,repi,fk,flarg,dkder,ddigam,ertolp
      double precision theta,ertolm

      common /DERRTOL/ ertolp,ertolm

      ic = (0.d0,1.d0)
      f = cdfact(m,arg)
      fm = dble(m)
      gamma = 0.57721566490153d0
      pi = 3.1415926535897932d0
      repi = 1.0d0/pi

      if (zabs(arg) .lt. 1.5*ertolm) then
        dbesy1 = ertolp
        return
      endif

      emz = 2.0d0*repi*(gamma + zlog(0.5d0*arg))
      call dcdbsj1(arg,m,dbesj1)
      demzj1 = emz*dbesj1
      call dcbsj1(arg,m,besj1)
      demzj2 = besj1*2.0d0*repi/arg

      if (m .eq. 0) then
        dfmz = 0.0d0
      else if (m .eq. 1) then
        dfmz = 2.0d0*repi/arg**2
      else
        if (abs(arg*f) .lt. .5*ertolm) then
          qntdb = ertolp
        else
          qntdb = 1.d0/(.5d0*arg*f)
        endif
        k = 1
10      kmm = m - 1 - k
        fk = dble(k)
        qntdb = qntdb + ((fm-(2.d0*fk))*cdfact(k,arg)) /
     .               (cdfact(kmm,arg)*(.5d0*arg)**2)
        if (zabs(qntdb) .lt. 1.5d0*ertolm) then
          dfmz = 0.0d0
        else if (zabs(qntdb) .gt. .5d0*ertolp) then
          theta = datan(dimag(qntdb)/dble(qntdb))
          dfmz = ertolp*exp(ic*fm*theta)
```

```
        else
          dfmz = .5d0*qntdb*repi
        endif
        k = k + 1
        if (k .le. m-1) go to 10
      endif

      if (zabs(f) .lt. 1.5d0*ertolm) then
        dergmz = 0.0d0
      else if (zabs(f) .gt. .5d0*ertolp) then
          theta = datan(dimag(f)/dble(f))
        dergmz = ertolp*exp(ic*fm*theta)
      else
        larg = int(8.0d0+1.4d0*zabs(arg))
        qntder = 1.0d0
        do 20 k = 1,larg
          klarg = larg - k + 1
          flarg = dble(larg-k+1)
          dkder = (ddigam(klarg+1) + ddigam(m+klarg+1)) *
     .              (fm+2.d0*flarg+2.d0)
     .            / ( (ddigam(klarg)+ddigam(m+klarg))*(flarg +1.d0)*
     .                (fm+flarg +1.d0)*(fm+2.d0*flarg) )
          qntder = 1.0d0 - qntder*dkder*(0.5d0*arg)**2
20      continue
        sumder = fm*ddigam(m) - qntder*(1.d0+ddigam(m+1))*
     .           (fm + 2.d0)*(0.5d0*arg)**2/(fm+1.d0)
        dergmz = -0.5d0*repi*f*sumder/(0.5d0*arg)
      endif

      dbesy1 = demzj1 + demzj2 + dfmz + dergmz
      call cdtolch(dbesy1,dbesy1,ertolp,0.0d0,iflag)
      return
      end

c*******************************************
      subroutine dcbesjy3(arg,m,icode,besj3,besy3,dbesj3,dbesy3)

c   Airy function approximation to Bessel function, complex argument

      integer m,icode,lmk,k
      double complex arg,besj3,besy3,dbesj3,dbesy3,srarg,angle
      double complex qnta,qntb,umk,vmk,qntad,qntbd,rmk,smk
      double precision ertolp,ertolm,fm,pi,fmsr,flmk4,amk,bmk,amkd,bmkd
      double precision theta

      common /DERRTOL/  ertolp,ertolm

      fm = dble(m)
      pi = 3.1415926535897932d0
      srarg = sqrt(2.0d0/(pi*arg))
```

```
      angle = arg - 0.25d0*pi*(1.0d0 + 2.0d0*fm)
      fmsr = 4.0d0*fm**2
      lmk = int( 9.d0 + zabs(arg)*dexp(-.0458d0*zabs(arg)) )
      if (lmk .lt. 1) lmk = 1

      qnta = 1.0d0
      qntb = 1.0d0
      do 10 k = 1,lmk
        flmk4 = 4.0d0*dble(lmk-k+1)
        amk = (4.d0*(fmsr-(flmk4-3.d0)**2)*(fmsr-(flmk4-1.d0)**2)) /
     .        (flmk4*(flmk4 - 2.d0))
        bmk = (4.d0*(fmsr-(flmk4-1.d0)**2)*(fmsr-(flmk4+1.d0)**2)) /
     .        (flmk4*(flmk4 + 2.d0))
        qnta = 1.d0 - qnta*amk/(8.d0*arg)**2
        qntb = 1.d0 - qntb*bmk/(8.d0*arg)**2
        if (zabs(qnta) .ge. .5d0*ertolp) then
          theta = datan(dimag(qnta)/dble(qnta))
          qnta = ertolp*exp((0.d0,1.d0)*theta)
        endif
        if (zabs(qntb) .ge. .5d0*ertolp) then
          theta = datan(dimag(qntb)/dble(qntb))
          qntb = ertolp*exp((0.d0,1.d0)*theta)
        endif
10    continue
      umk = qnta
      vmk = qntb*(fmsr - 1.0d0)/(8.0d0*arg)
      besj3 = srarg*(umk*cos(angle) - vmk*sin(angle))
      if (zabs(besj3) .ge. .5d0*ertolp) then
        theta = datan(dimag(besj3)/dble(besj3))
        besj3 = ertolp*exp((0.d0,1.d0)*theta)
      endif
      besy3= srarg*(umk*sin(angle) + vmk*cos(angle))
      if (zabs(besy3) .ge. .5d0*ertolp) then
        theta = datan(dimag(besy3)/dble(besy3))
        besy3 = ertolp*exp((0.d0,1.d0)*theta)
      endif

      if (icode .eq. 3) then
        qntad = 1.0d0
        qntbd = 1.0d0
        do 20 k = 1,lmk
          flmk4 = 4.0d0*dble(lmk-k+1)
          amkd = (4.d0*(fmsr+(flmk4**2-1.d0))*(fmsr-(flmk4-3.d0)**2)*
     .            (fmsr-(flmk4-5.d0)**2)) /
     .            (flmk4*(flmk4 - 2.d0)*(fmsr+(flmk4-4.d0)**2-1.d0))
          bmkd = (4.d0*(fmsr+(flmk4+2.d0)**2-1.d0)*
     .            (fmsr-(flmk4-3.d0)**2)*(fmsr-(flmk4-1.d0)**2)) /
     .            (flmk4*(flmk4 + 2.d0)*(fmsr+(flmk4-2.d0)**2 - 1.d0))
          qntad = 1.d0 - qntad*amkd/(8.d0*arg)**2
          qntbd = 1.d0 - qntbd*bmkd/(8.d0*arg)**2
```

```
            if (zabs(qntad) .ge. .5d0*ertolp) then
               theta = datan(dimag(qntad)/dble(qntad))
               qntad = ertolp*exp((0.d0,1.d0)*theta)
            endif
            if (zabs(qntbd) .ge. .5d0*ertolp) then
               theta = datan(dimag(qntbd)/dble(qntbd))
               qntbd = ertolp*exp((0.d0,1.d0)*theta)
            endif
20        continue
          rmk = qntad
          smk = qntbd*(fmsr + 3.0d0)/(8.0d0*arg)
          dbesj3 = -srarg*(rmk*sin(angle) + smk*cos(angle))
          if (zabs(dbesj3) .ge. .5d0*ertolp) then
             theta = datan(dimag(dbesj3)/dble(dbesj3))
             dbesj3 = ertolp*exp((0.d0,1.d0)*theta)
          endif
          dbesy3 = srarg*(rmk*cos(angle) - smk*sin(angle))
          if (zabs(dbesy3) .ge. .5d0*ertolp) then
             theta = datan(dimag(dbesy3)/dble(dbesy3))
             dbesy3 = ertolp*exp((0.d0,1.d0)*theta)
          endif
       endif

       return
       end

c*******************************************
       subroutine dcbesjy2(arg,m,icode,besj2,besy2,dbesj2,dbesy2)

c   asymptotic approximation to Bessel function, complex argument
c     Abr-Stegun 9.3.35, 9.3.36

       integer m,icode
       double complex arg,besj2,besy2,dbesj2,dbesy2,x,zeta,q1
       double complex q1i,g3,g32,g2,gsq,a1,b0,c0,d1,h,y,airy
       double complex d,ccos,phi,f1,f2,f3,biry,dairy,dbiry,ic
       double complex u1,u2,u3,b1,qq,u4,u5,a2,b2
       double precision ertolp,ertolm,fm,q,w,a,b

       common /DERRTOL/ ertolp,ertolm

       ic = (0.d0,1.d0)
       fm = dble(m)
       x = arg/fm
       if (zabs(x-1.d0) .lt. 1.5d0*ertolm) then
          zeta = 0.0d0
       else if (dble(x) .gt. 1.d0) then
          q = dble(x) / (dble(x)**2+dimag(x)**2)
          w = -dimag(x) / (dble(x)**2+dimag(x)**2)
c         q + iw = 1/x
```

```
      a = .5d0*( sqrt((q+1.d0)**2+w**2) + sqrt((q-1.d0)**2+w**2) )
      b = .5d0*( sqrt((q+1.d0)**2+w**2) - sqrt((q-1.d0)**2+w**2) )
      d = a + sqrt(a**2-1.d0)
      ccos = acos(b) - ic*zlog(d)
      if (dimag(x) .gt. 0.d0) ccos = dble(ccos)-ic*dimag(ccos)
      zeta = -(1.5d0*(sqrt(x**2-1.d0) - ccos))**(2.d0/3.d0)
    else
      zeta = (1.5d0*(zlog((1.0d0 +sqrt(1.d0-x**2))/x)
     .        - sqrt(1.d0-x**2)))**(2.d0/3.d0)
    endif

    if (zabs(x-1.d0) .lt. 0.02d0) then
      h = 1.0d0 - x
      phi = 2.0d0**(1.d0/3.d0)*(1.0d0 + 0.2d0*h + 3.d0*h**2/35.d0
     .      + 73.d0*h**3/1575.d0 + 35209.d0*h**4/1212750.d0 +
     .        380069.d0*h**5/18768750.d0)
      a1 = -1.d0/225.d0 - 71.d0*h/38500.d0
      b0 = (1.d0/70.d0 + 2.d0*h/225.d0)*2**(1.d0/3.d0)
      b1 = 2.d0**(1.d0/3.)*(-1213.d0/1023750.d0 - 3757.d0*h/2695000.
     .      - h**2*(8.9962899979797d-4 + h*(.0002753433716d0 - h*
     .      (.00018048868d0 + h*.0004108523))))
      a2 = 6.937355413546877d-4 + h*(.00046448349036601 - h*
     .      (.0002890362546053d0 + h*(.0008747649439535d0 +
     .      h*.00102971637614)))
      b2 = -2.d0**(1./3.)*(4.382918094497229d-4 + h*
     .      (7.1104865116911d-4 + h*5.318984348085d-4))
      b2 = b2 - 2.d0**(1./3.)*h**3*2.182958472d-4
      c0 = (0.1d0 + 0.02d0*h)*2**(2.d0/3.d0)
      d1 = 23.d0/3150.d0 + 1453.d0*h/346500.d0
    else
      q1 = zeta/(1.0d0-x**2)
      phi = (4.0d0*q1)**0.25d0
      f1 = 1.0d0 - x**2
      f2 = f1**2
      f3 = f1**3
      q1i = 1.0d0/q1
      g3 = q1i**3
      g32 = q1i**1.5d0
      g2 =  q1i**2
      gsq = sqrt (q1i)
      a1 = (-455.d0*g3/4608.d0-7.d0*g32*(f1-5.d0/3.d0)/384.d0+385.d0/
     .      1152.d0 - 77.d0*f1/192.d0 + 9.d0*f2/128.d0 ) / f3
      b0 = (-5.d0*g2/48.d0 + gsq*(5.d0/24.d0 - f1/8.d0))/f2
      c0 = (7.d0*q1i/48.d0 - 3.d0*sqrt(q1)*(7.d0/72.d0 - f1/8.d0))/f1
      d1 = (385.d0*g3/4608.d0+5.d0*g32*(-f1+7.d0/9.d0)/128.d0
     .      -15.d0*f2/128.d0 + 33.d0*f1/64.d0 - 455.d0/1152.d0 ) / f3
      qq = sqrt(f1/zeta)
      u1 = qq*(1.d0-5.d0/(f1*3.d0)) / (8.d0*f1)
      u2 = (4.5d0 - 77.d0/(f1*3.d0)*(1.d0-5.d0/(f1*6.d0))) /
     .      (64.d0*f1)
```

```
      u3 = qq*(75.d0/2. - 456.3d0/f1 + 17017.d0/(f2*18.d0)*(1.d0-
     .         5.d0/(f1*9.d0))) / (512.d0*f2)
      u4 = (3675.d0/8.d0 - 9683.3d0/f1 + 2717.d0/f2*(53.d0/4.d0 -
     .      2737.d0/(f1*162.d0)*(1.d0-5.d0/(12.d0*f1))))/(4096.d0*f2)
      u5 = 59535.d0/8.d0 - 221.d0/(4.d0*f1)*(305923.d0/70.d0 -
     .      77.d0/f1*(14743.d0/45.d0 - 95.d0/f1*(67.d0/9.d0-
     .      3335.d0/(486.d0*f1)*(1.d0-1.d0/(3.d0*f1)))))
      u5 = qq*u5/(f3*32768.d0)
      b1 = -u3 - 5.d0/(zeta**2*48.d0)*(u2+77.d0/(zeta*96.d0)*
     .      (u1 + 221.d0/(zeta**2*144.d0)))
      b2 = -u5 - 5.d0/(zeta**2*48.d0)*(u4 + 77.d0/(zeta*96.d0)*
     .      (u3 + 221.d0/(zeta**2*144.d0)*(u2 + 437.d0/(zeta*192.d0)*
     .      (u1 + 145.d0/(zeta**2*48.d0)))))
      a2 = u4 - 7.d0/(zeta*48.d0)*(u3 + 65.d0/(zeta**2*96.d0)*(u2 +
     .      209.d0/(zeta*144.d0)*(u1 + 425.d0/(zeta**2*192.d0))))
    endif

    y = zeta*fm**(2.d0/3.d0)
    call cdairyfn(y,airy,biry,dairy,dbiry)

    if (icode .eq. 1) then
      besj2 = phi/(fm**(1.d0/3.d0))*(airy*(1.d0+a1/fm**2+a2/fm**4)
     .         + dairy / (fm**(4.d0/3.d0))*(b0+b1/fm**2+b2/fm**4))
    else if (icode .eq. 2) then
      besy2 = -phi/(fm**(1.d0/3.d0))*(biry*(1.d0+a1/fm**2+a2/fm**4)
     .         + dbiry / (fm**(4.d0/3.d0))*(b0+b1/fm**2+b2/fm**4))
      call cdtolch(besy2,besy2,ertolp,0.0d0,iflag)
    else
      besj2 = phi/(fm**(1.d0/3.d0))*(airy*(1.d0+a1/fm**2+a2/fm**4)
     .         + dairy / (fm**(4.d0/3.d0))*(b0+b1/fm**2+b2/fm**4))
      besy2 = -phi/(fm**(1.d0/3.d0))*(biry*(1.d0+a1/fm**2+a2/fm**4)
     .         + dbiry / (fm**(4.d0/3.d0))*(b0+b1/fm**2+b2/fm**4))
      call cdtolch(besy2,besy2,ertolp,0.0d0,iflag)
      dbesj2 = -2.d0/( fm**(2.d0/3.d0)*x*phi)*(airy/
     .          (fm**(2.d0/3.d0))*c0 + dairy*(1.d0 + d1/(fm**2.d0)))
      dbesy2 = 2.d0/(fm**(2.d0/3.d0)*x*phi)*(biry*c0/
     .          (fm**(2.d0/3.d0))+ dbiry*(1.d0 + d1/(fm**2.d0)))
      call cdtolch(dbesy2,dbesy2,ertolp,0.0d0,iflag)
    endif

    return
    end

c*****************************************
    subroutine cdairyfn(y,airy,biry,dairy,dbiry)

    integer icode,l1,k,k2,l3,l4,l5,lp,lpt
    double complex y,airy,biry,dairy,dbiry,flam,reflam
    double complex clam,srtwo,qntv,ffm1,vffm1,vffmr,qntw
    double complex qntdv,ffm3,dvffmr,qntdw,ffm4,dwffmr,sum1
```

```
      double complex sum4,dift1,sumt1,dift2,sumt2,qntf,qntg
      double complex ff,fg,fdf,fdg,fml,fpl,fmp,gml,gpl,fmpt
      double complex a1,b1,da1,db1,a2,b2,da2,db2,a3,b3,da3,db3
      double complex aby,slam,ffm2,wffmr,sum2,sum3,qntdf
      double complex qntdg,sxfmpt,d12,d13,sxfmp
      double precision ertolp,ertolm,c1,c2,ffm5,err,r12,r13,pi,srpi

      common /DERRTOL/ ertolp,ertolm

      c1 = 0.355028053887817d0
      c2 = 0.258819403792807d0

      if (zabs(y) .lt. ertolm) then

c        Ai(0) etc from Abr-Stegun 10.4.4, 10.4.5
         airy = c1
         biry = c1*sqrt(3.d0)
         dairy = -c2
         dbiry = sqrt(3.d0)*c2

      else if (zabs(y) .ge. 25.d0) then

c        at machine tolerance
         airy = 0.0d0
         dairy = 0.0d0
         biry = ertolp
         dbiry = ertolp

      else if (zabs(y) .lt. 4.0d0) then

c        power series
         L5  = int(5.0d0 + 3.5d0*zabs(y))
         qntf = 1.0d0
         qntg = 1.0d0
         qntdf = 1.0d0
         qntdg = 1.0d0
         do 84 k = 1,L5
            ffm5 = 3.d0*dble(L5-k+1)
            qntf = 1.0d0 + qntf*y**3/(ffm5*(ffm5 - 1.0d0))
            qntg = 1.0d0 + qntg*y**3/(ffm5*(ffm5 + 1.0d0))
            qntdf = 1.0d0 + qntdf*y**3/(ffm5*(ffm5 + 2.0d0))
            qntdg = 1.0d0 + qntdg*y**3/(ffm5*(ffm5-2.0d0))
84       continue
         ff = qntf
         fg = qntg*y
         fdf = qntdf*0.5d0*y**2
         fdg = qntdg
         airy = c1*ff - c2*fg
         dairy = c1*fdf - c2*fdg
         biry = sqrt(3.0d0)*(c1*ff + c2*fg)
```

```
         dbiry = sqrt(3.0d0)*(c1*fdf + c2*fdg)
         call cdtolch(airy,airy,ertolp,0.0d0,iflag)
         call cdtolch(biry,biry,ertolp,0.0d0,iflag)
         call cdtolch(dairy,dairy,ertolp,0.0d0,iflag)
         call cdtolch(dbiry,dbiry,ertolp,0.0d0,iflag)

      else if ( (dble(y) .ge. 0.0d0) .or.
     .       (dabs(dimag(y)/dble(y)) .ge. sqrt(3.d0)/2.d0) ) then

c        exponential approx (Abr-Stegun 10.4.59)
         flam = (2.d0/3.d0)*(y)**1.5d0
         reflam = 1.0d0/flam
         pi = 3.1415926535897932d0
         srpi = dsqrt(pi)
         aby = (y)**0.25d0
         lp = int(5.d0 + 70.0d0*zabs(reflam))
         fml = 1.0d0
         fpl = 1.0d0
         do 115 k = 1,lp
            sxfmp = 6.0d0*dble(lp-k+1)
            fml = 1.d0-fml*reflam*(sxfmp-5.d0)*(sxfmp-1.d0)/
     .            (12.d0*sxfmp)
            fpl = 1.d0+fpl*reflam*(sxfmp-5.d0)*(sxfmp-1.d0)/
     .            (12.d0*sxfmp)
115      continue
         lpt = int(4.0d0 + 90.0d0*zabs(reflam))
         gml = 1.0d0
         gpl = 1.0d0
         do 125 k = 1,lpt
            sxfmpt = 6.0d0*dble(lpt-k+1)
            gml = 1.d0-gml*reflam*(sxfmpt+1.d0)*(sxfmpt-7.d0)/
     .            (22.d0*sxfmpt)
            gpl = 1.d0+gpl*reflam*(sxfmpt+1.d0)*(sxfmpt-7.d0)/
     .            (12.d0*sxfmpt)
125      continue
         airy = 0.5d0*y**(-0.25d0)*fml*exp(-flam)/srpi
         dairy = -0.5d0*y**0.25d0*gml*exp(-flam)/srpi
         biry = y**(-0.25d0)*fpl*exp(flam)/srpi
         dbiry =y**0.25d0*gpl*exp(flam)/srpi
         call cdtolch(airy,airy,ertolp,0.0d0,iflag)
         call cdtolch(biry,biry,ertolp,0.0d0,iflag)
         call cdtolch(dairy,dairy,ertolp,0.0d0,iflag)
         call cdtolch(dbiry,dbiry,ertolp,0.0d0,iflag)

      else

c        trig approx (Abr-Stegun 10.4.60)
         flam = (2.d0/3.d0)*(-y)**1.5d0
         reflam = 1.0d0/flam
         aby = (-y)**0.25d0
```

```
      slam = sin(flam)
      clam = cos(flam)
      pi = 3.1415926535897932d0
      srpi = dsqrt(pi)
      srtwo = sqrt(2.d0)/2.d0
      L1  = int(1.0d0 + 75.0d0*zabs(reflam))

      qntv = 1.0d0
      do 20 k = 1,L1
        ffm1 = 12.d0 *dble(L1-k+1)
        vffmr = (ffm1-11.d0)*(ffm1-7.d0)*(ffm1-5.d0)*(ffm1-1.d0)
     .        / (144.d0*ffm1*(ffm1-6.d0))
        qntv = 1.0d0 - (qntv*vffmr*reflam**2.d0)
20     continue
      L2  = int(1.0d0 + 70.0d0*zabs(reflam))
      qntw = 1.0d0
      do 30 k = 1,L2
        ffm2 = 12.d0*dble(L2-k+1)
        wffmr = (ffm2-5.d0)*(ffm2-1.d0)*(ffm2+1.d0)*(ffm2+5.d0)
     .        / (144.d0*ffm2*(ffm2 + 6.d0))
        qntw = 1.0d0 - (qntw*wffmr*reflam**2.d0)
30     continue
      L3  = int(1.0d0 + 60.0d0*zabs(reflam))
      qntdv = 1.0d0
      do 40 k = 1,L3
        ffm3 = 12.d0*dble(L3-k+1)
        dvffmr = (ffm3-7.d0)*(ffm3-1.d0)*(ffm3+1.d0)*(ffm3+7.d0)
     .        / (144.d0*ffm3*(ffm3 + 6.d0))
        qntdv = 1.0d0 - (qntdv*dvffmr*reflam**2.d0)
40     continue
      L4  = int(1.0d0 + 33.0d0*zabs(reflam))
      qntdw = 1.0d0
      do 50 k = 1,L4
        ffm4 = 12.d0*dble(L4-k+1)
        dwffmr = (ffm4-13.d0)*(ffm4-7.d0)*(ffm4-5.d0)*(ffm4+1.d0)
     .        / (144.d0*ffm4*(ffm4 - 6.d0))
        qntdw = 1.0d0 - (qntdw*dwffmr*reflam**2.d0)
50     continue
      sum1 = qntv/srpi
      sum2 = 5.d0*qntw*reflam/(72.d0*srpi)
      sum3 = -7.d0*qntdv*reflam/(72.d0*srpi)
      sum4 = qntdw/srpi
      dift1 = srtwo*(sum1 - sum2)
      sumt1 = srtwo*(sum1 + sum2)
      dift2 = srtwo*(sum4 - sum3)
      sumt2 = srtwo*(sum4+sum3)

      airy = (dift1*clam + sumt1*slam)/aby
      dairy = (dift2*slam - sumt2*clam)*aby
      biry = (sumt1*clam - dift1*slam)/aby
```

```
         dbiry = (sumt2*slam + dift2*clam)*aby
         call cdtolch(airy,airy,ertolp,0.0d0,iflag)
         call cdtolch(biry,biry,ertolp,0.0d0,iflag)
         call cdtolch(dairy,dairy,ertolp,0.0d0,iflag)
         call cdtolch(dbiry,dbiry,ertolp,0.0d0,iflag)

      endif

      return
      end

c****************************************************
      subroutine cdtolch(cin,cout,setbig,setsmall,iflag)

c   complex, double precision check to avoid over- and underflow

      integer iflag
      double precision setbig,setsmall,theta,pi,ertolp,ertolm
      double complex cin,cout,ic

      common /DERRTOL/  ertolp,ertolm

      pi = 3.1415926535897932d0
      ic = (0.d0,1.d0)
      iflag = 0
      if (zabs(cin) .lt. 1.5d0*ertolm) then
         theta = datan(dimag(cin)/dble(cin))
         if (dble(cin) .lt. 0.0d0) theta = theta + pi
         cout = setsmall*exp(ic*theta)
         iflag = 1
      else if (zabs(cin) .gt. .5d0*ertolp) then
         theta = datan(dimag(cin)/dble(cin))
         if (dble(cin) .lt. 0.0d0) theta = theta + pi
         cout = setbig*exp(ic*theta)
         iflag = 1
      endif
      return
      end
```

REFERENCES

1. Abramowitz, M.; and Stegun, I., eds.: Handbook of Mathematical Functions. National Bureau of Standards, Applied Mathematical Series 55, 1964.

2. Hildebrand, F.: Advanced Calculus for Engineers. Prentice-Hall, 1948.

TABLE I - SINGLE PRECISION, REAL ARGUMENT

m	x	$J_m(x)$	$Y_m(x)$	$J'_m(x)$	$Y'_m(x)$
1	10.0	4.3472745×10^{-2}	0.2490154	-0.2502830	3.0769626×10^{-2}
10	2.0	2.5153864×10^{-7}	-129184.5	1.2346503×10^{-6}	631362.9
20	25.0	5.1994048×10^{-2}	0.1980408	-0.1230286	2.1158613×10^{-2}

TABLE II - DOUBLE PRECISION, REAL ARGUMENT

m	x	$J_m(x)$	$Y_m(x)$	$J'_m(x)$	$Y'_m(x)$
1	10.0	$4.3472746168805587 \times 10^{-2}$	0.2490154242067848	-0.2502830390682086	$3.0769624863030031 \times 10^{-2}$
10	2.0	$2.5153862827167365 \times 10^{-7}$	-129184.5422080393	1.2346503×10^{-6}	$1.2346502937746957 \times 10^{-6}$
20	25.0	$5.1994049228302969 \times 10^{-2}$	0.1980407477628901	-0.123-285643023131	$2.1158614118514424 \times 10^{-2}$

TABLE III - SINGLE PRECISION, COMPLEX ARGUMENT

m	r	θ, degrees	$J_m(x)$	$Y_m(x)$	$J'_m(x)$	$Y'_m(x)$
1	10.0	37	48.46594, 14.90409	-14.90460,48.46558	12.67906, -47.47811	47.47851.12.67855
10	2.0	-15	$-2.1449186 \times 10^{-7}$, $-1.3728736 \times 10^{-7}$	1064829.7, -69704.96	$-8.3601503 \times 10^{-7}$, $-9.3365060 \times 10^{-7}$	-594765.0, 188178.8
20	25.0	80	1309767., 1275952.	$-4.3071617 \times 10^{-9}$, 3.3826593×10^{-9}	1714659., -1538981.	$-3.9587404 \times 10^{-9}$, $5,8104996 \times 10^{-9}$

TABLE IV - DOUBLE PRECISION, COMPLEX ARGUMENT

m	r	θ, degrees	$J_m(x)$	$Y_m(x)$
1	10.0	37	48.46594122869320, 14.90408497152226	-14.90459530350984, 48.46557450542743
10	2.0	-15	$-2.1449186196047720 \times 10^{-7}$, $-1.3728736108270844 \times 10^{-7}$	$-8.3601502603410100 \times 10^{-7}$, $-9.3365061711294054 \times 10^{-7}$
20	25.0	80	1309766.766321728,1275951.977909705	1714659.236311793, -1538981.266957863

Double Precision, Complex Argument

m	r	θ, degrees	$J'_m(x)$	$Y'_m(x)$
1	10.0	37	12.67906150921442, -47.47811148316850	47.47851111589829, 12.67855209985360
10	2.0	-15	$-8.3061502603410100 \times 10^{-7}$, $-9.3365061711294057 \times 10^{-7}$	-594764.9878571380,188178.8391119776
20	25.0	80	1714659.236311793, -1538981.266957863	$-3.9587403354834531 \times 10^{-9}$, $5.8104997303197833 \times 10^{-9}$

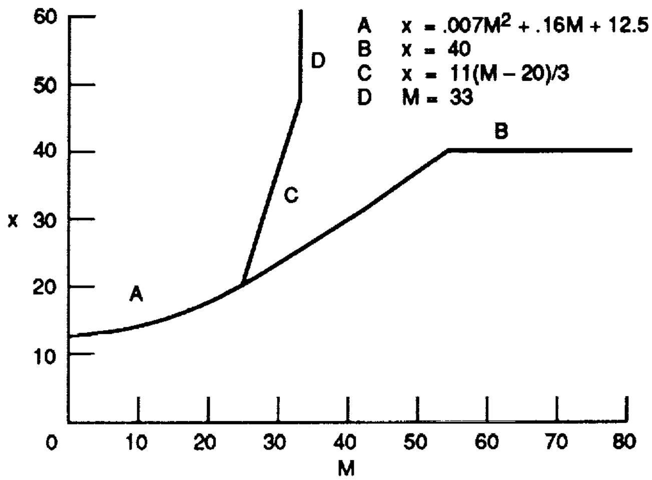

Figure 1.—Evaluation regions for Bessel functions with real argument.

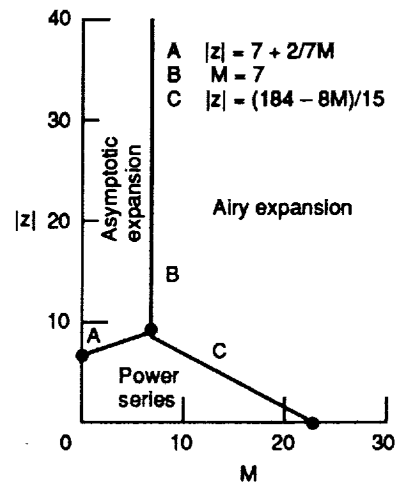

Figure 2.—Evaluation regions for Bessel functions with complex argument.

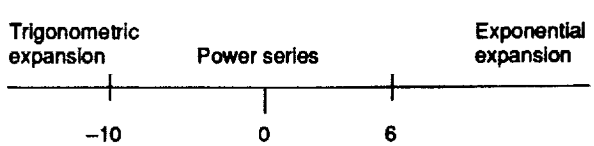

Figure 3.—Evaluation regions for Airy functions with real argument.

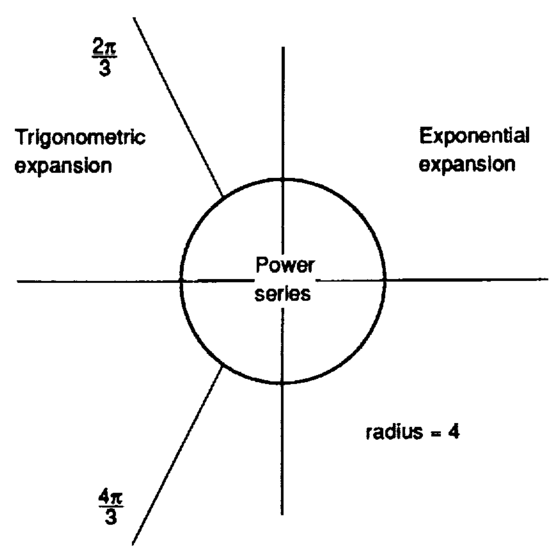

Figure 4.—Evaluation regions for Airy functions with complex argument.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
October 1991

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE
Computer Program for Bessel and Hankel Functions

5. FUNDING NUMBERS
505-62-21

6. AUTHOR(S)
Kevin L. Kreider, Arthur V. Saule, Edward J. Rice, and Bruce J. Clark

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E-6439

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-105154

11. SUPPLEMENTARY NOTES
Kevin L. Kreider, University of Akron, Department of Mathematical Sciences, Akron, Ohio 44325; Arthur V. Saule (retired), Edward J. Rice, and Bruce J. Clark, NASA Lewis Research Center. Responsible person, Edward J. Rice, (216) 433-5885.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 61

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***
Bessel and Hankel functions are widely used in many research and application areas. A set of FORTRAN subroutines for calculating these functions is presented. The routines calculate Bessel and Hankel functions of the first and second kinds, as well as their derivatives, for wide ranges of integer order and real or complex argument in single or double precision. Depending on the order and argument, one of three evaluation methods is used: the power series definition, an Airy function expansion, or an asymptotic expansion. Routines to calculate Airy functions and their derivatives are also included.

14. SUBJECT TERMS
Bessel functions; Hankel functions; Special functions; Airy functions

15. NUMBER OF PAGES
66

16. PRICE CODE
A04

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	