6.7 Bessel Functions of Fractional Order, Airy Functions, Spherical Bessel Functions

Many algorithms have been proposed for computing Bessel functions of fractional order numerically. Most of them are, in fact, not very good in practice. The routines given here are rather complicated, but they can be recommended wholeheartedly.

**Ordinary Bessel Functions**

The basic idea is Steed's method, which was originally developed [1] for Coulomb wave functions. The method calculates \( J_\nu, J'_\nu, Y_\nu, \) and \( Y'_\nu \) simultaneously, and so involves four relations among these functions. Three of the relations come from two continued fractions, one of which is complex. The fourth is provided by the Wronskian relation

\[
W = J_\nu Y'_\nu - Y_\nu J'_\nu = \frac{2}{\pi x}.
\]

The first continued fraction, CF1, is defined by

\[
f_\nu \equiv \frac{J'_\nu}{J_\nu} = \frac{\nu}{x} - \frac{1}{2(\nu + 1)/x - 2(\nu + 2)/x - \cdots}.
\]

You can easily derive it from the three-term recurrence relation for Bessel functions: Start with equation (6.5.6) and use equation (5.5.18). Forward evaluation of the continued fraction by one of the methods of §5.2 is essentially equivalent to backward recurrence of the recurrence relation. The rate of convergence of CF1 is determined by the position of the turning point

\[x_{tp} = \sqrt{\nu(\nu + 1)} \approx \nu,
\]

beyond which the Bessel functions become oscillatory. If \( x < x_{tp} \), convergence is very rapid. If \( x > x_{tp} \), then each iteration of the continued fraction effectively increases \( \nu \) by one until \( x < x_{tp} \); thereafter rapid convergence sets in. Thus the number of iterations of CF1 is of order \( x \) for large \( x \). In the routine `bessjy` we set the maximum allowed number of iterations to 10,000. For larger \( x \), you can use the usual asymptotic expressions for Bessel functions.

One can show that the sign of \( J_\nu \) is the same as the sign of the denominator of CF1 once it has converged.

The complex continued fraction CF2 is defined by

\[
p + iq \equiv \frac{J'_\nu + iY'_\nu}{J_\nu + iY_\nu} = \frac{1}{2x} + i \frac{1}{2x(2x + i) + \cdots}.
\]

(We sketch the derivation of CF2 in the analogous case of modified Bessel functions in the next subsection.) This continued fraction converges rapidly for \( x > x_{tp} \), while convergence fails as \( x \to 0 \). We have to adopt a special method for small \( x \), which we describe below. For \( x \) not too small, we can ensure that \( x > x_{tp} \), by a stable recurrence of \( J_\nu \) and \( J'_\nu \) downwards to a value \( \nu = \mu \approx x \), thus yielding the ratio \( f_\mu \) at this lower value of \( \nu \). This is the stable direction for the recurrence relation. The initial values for the recurrence are

\[
J_\nu = \text{arbitrary}, \quad J'_\nu = f_\nu J_\nu,
\]

with the sign of the arbitrary initial value of \( J_\nu \) chosen to be the sign of the denominator of CF1. Choosing the initial value of \( J_\nu \) very small minimizes the possibility of overflow during the recurrence. The recurrence relations are

\[
J_{\nu-1} = \frac{\nu}{x} J_\nu + J'_\nu,
J'_\nu = \frac{\nu - 1}{x} J_{\nu-1} - J_\nu
\]
Once CF2 has been evaluated at $\nu = \mu$, then with the Wronskian (6.7.1) we have enough relations to solve for all four quantities. The formulas are simplified by introducing the quantity

$$\gamma \equiv \frac{p - f_\mu}{q}$$

Then

$$J_\mu = \pm \left( \frac{W}{q + \gamma(p - f_\mu)} \right)^{1/2}$$

$$J'_\mu = f_\mu J_\mu$$

$$Y_\mu = \gamma J_\mu$$

$$Y'_\mu = Y_\mu \left( p + \frac{q}{\gamma} \right)$$

(6.7.6) (6.7.7) (6.7.8) (6.7.9) (6.7.10)

The sign of $J_\mu$ in (6.7.7) is chosen to be the same as the sign of the initial $J_\nu$ in (6.7.4).

Once all four functions have been determined at the value $\nu = \mu$, we can find them at the original value of $\nu$. For $J_\nu$ and $J'_\nu$, simply scale the values in (6.7.4) by the ratio of (6.7.7) to the value found after applying the recurrence (6.7.5). The quantities $Y_\nu$ and $Y'_\nu$ can be found by starting with the values in (6.7.9) and (6.7.10) and using the stable upwards recurrence

$$Y_{\nu+1} = \frac{2\nu}{x} Y_\nu - Y_{\nu-1}$$

(6.7.11) together with the relation

$$Y'_\nu = \frac{\nu}{x} Y_\nu - Y_{\nu+1}$$

(6.7.12)

Now turn to the case of small $x$, when CF2 is not suitable. Temme [2] has given a good method of evaluating $Y_\nu$ and $Y_{\nu+1}$, and hence $Y'_\nu$ from (6.7.12), by series expansions that accurately handle the singularity as $x \to 0$. The expansions work only for $|\nu| \leq 1/2$, and so now the recurrence (6.7.5) is used to evaluate $f_\nu$ at a value $\nu = \mu$ in this interval. Then one calculates $J_\nu$ from

$$J_\nu = \frac{W}{Y'_\nu - Y_\nu f_\nu}$$

(6.7.13) and $J'_\nu$ from (6.7.8). The values at the original value of $\nu$ are determined by scaling as before, and the $Y$'s are recurred up as before.

Temme's series are

$$Y_\nu = -\sum_{k=0}^{\infty} c_k g_k \quad Y_{\nu+1} = -\frac{2}{x} \sum_{k=0}^{\infty} c_k h_k$$

(6.7.14)

Here

$$c_k = \frac{(-x^2/4)^k}{k!}$$

(6.7.15)

while the coefficients $g_k$ and $h_k$ are defined in terms of quantities $p_k$, $q_k$, and $f_k$ that can be found by recursion:

$$g_k = f_k + \frac{2}{\nu} \sin^2 \left( \frac{\nu \pi}{2} \right) q_k$$

$$h_k = -k g_k + p_k$$

$$p_k = \frac{p_{k-1}}{k - \nu}$$

$$q_k = \frac{q_{k+1}}{k + \nu}$$

$$f_k = \frac{f_{k-1} + p_{k-1} + q_{k-1}}{k^2 - \nu^2}$$

(6.7.16)
The initial values for the recurrences are
\[
\begin{align*}
p_0 &= \frac{1}{\pi} \left( \frac{x}{2} \right)^\nu \Gamma(1 + \nu) \\
q_0 &= \frac{1}{\pi} \left( \frac{x}{2} \right)^\nu \Gamma(1 - \nu) \\
f_0 &= \frac{2}{\pi} \frac{\nu \pi}{\sin \nu \pi} \left[ \cosh \sigma \Gamma_1(\nu) + \sinh \sigma \ln \left( \frac{2}{x} \right) \Gamma_2(\nu) \right]
\end{align*}
\] (6.7.17)

with
\[
\sigma = \nu \ln \left( \frac{2}{x} \right)
\]
\[
\Gamma_1(\nu) = \frac{1}{2} \left[ \frac{1}{\Gamma(1 - \nu)} - \frac{1}{\Gamma(1 + \nu)} \right]
\]
\[
\Gamma_2(\nu) = \frac{1}{2} \left[ \frac{1}{\Gamma(1 - \nu)} + \frac{1}{\Gamma(1 + \nu)} \right]
\] (6.7.18)

The whole point of writing the formulas in this way is that the potential problems as \( \nu \to 0 \) can be controlled by evaluating \( \nu \pi / \sin \nu \pi, \sinh \sigma / \sigma \), and \( \Gamma_1 \) carefully. In particular, Temme gives Chebyshev expansions for \( \Gamma_1(\nu) \) and \( \Gamma_2(\nu) \). We have rearranged his expansion for \( \Gamma_1 \) to be explicitly an even series in \( \nu \) so that we can use our routine chebev as explained in §5.8.

The routine assumes \( \nu \geq 0 \). For negative \( \nu \) you can use the reflection formulas
\[
J_{-\nu} = \cos \nu \pi J_\nu - \sin \nu \pi Y_\nu \\
Y_{-\nu} = \sin \nu \pi J_\nu + \cos \nu \pi Y_\nu
\] (6.7.19)

The routine also assumes \( x > 0 \). For \( x < 0 \) the functions are in general complex, but expressible in terms of functions with \( x > 0 \). For \( x = 0 \), \( Y_\nu \) is singular.

Internal arithmetic in the routine is carried out in double precision. To maintain portability, complex arithmetic has been recoded with real variables.

SUBROUTINE bessjy(x,xnu,rj,ry,rjp,ryp)
INTEGER MAXIT
REAL rj,rjp,ry,ryp,x,xnu,XMIN
DOUBLE PRECISION EPS,FPMIN,PI
PARAMETER (EPS=1.e-10,FPMIN=1.e-30,MAXIT=10000,XMIN=2.,
* PI=3.141592653589793d0)
C USES beschb

Returns the Bessel functions \( rj = J_\nu, ry = Y_\nu \) and their derivatives \( rjp = J'_\nu, ryp = Y'_\nu \), for positive \( x \) and for \( xnu = \nu \geq 0 \). The relative accuracy is within one or two significant digits of EPS, except near a zero of one of the functions, where EPS controls its absolute accuracy. FPMIN is a number close to the machine’s smallest floating-point number. All internal arithmetic is in double precision. To convert the entire routine to double precision, change the REAL declaration above and decrease EPS to \( 10^{-16} \). Also convert the subroutinebeschb.

INTEGER i,isign,l,nl
DOUBLE PRECISION a,b,br,bi,c,cr,ci,d,del,del1,den,di,dlr,dli,
* dr,ef,fact,fact2,fact3,ff,gam,gami,gam2,gammi,gampl,h,
* p,pimu,pimu2,q,rj1,rjmu1,rjpl1,rjtemp,ry1,
* rymu,rymup,rytemp,sum,sum1,temp,v,x2,xi,xi2,xmu,xmu2
if(x.le.0.or.xnu.lt.0.) pause 'bad arguments in bessjy'
if(x.lt.XMIN)then
\[ nl = \text{number of downward recurrences of the } J's \] and
\[ nl = \text{number of upward recurrences of the } Y's. \] xmu lies between \(-1/2\) and
\[ \text{else } \]
\[ nl = \text{the turning point for } x \geq XMIN. \]
endif
xmu=xnu-nl
xmu2=xmu+1.5d0
x2=1.5d0/x
xi=1.4d0/xi
w=xi2/PI
The Wronskian.
6.7 Bessel Functions of Fractional Order

Evaluate CF1 by modified Lentz's method (§5.2). isign keeps track of sign changes in the denominator.

```fortran
isign=1  h=xmu*xi
if (h.lt.FPMIN) h=FPMIN
b=x12*xmu
d=0. d0

do i=1,MAXIT
  b=b*x12
  d=b-d
  if (abs(d).lt.FPMIN) d=FPMIN
  c=-1. d0/c
  if (abs(c).lt.FPMIN) c=FPMIN
  del=c+d
  h=del*h
  if (d.lt.0. d0) isign=-isign
  if (abs(del-1. d0).lt.EPS) goto 1

enddo

if (isign.eq.0. d0) rjl=EPS

rjl=isign*FPMIN  ! Initialize J_\nu and J_\nu' for downward recurrence.
rjpl=b*rjl
rjl=rjpl
fact=xmu*xi

do l=n1,1,-1
  rjtemp=fact*rjl+rjpl
  fact=fact-xi
  rjpl=fact*rjtemp-rjl
  rjl=rjtemp
enddo

if (rjl.eq.0. d0) rjl=EPS

if (x.lt.XMIN) then
  Use series.
  x2=.5d0*x
  pimu=PI*xmu
  if (abs(pimu).lt.EPS) then
    fact=1.d0
  else
    fact=pimu/sin(pimu)
  endif
  d=-log(x2)
  e=xmu*d
  if (abs(e).lt.EPS) then
    fact2=1.d0
  else
    fact2=sinh(e)/e
  endif
  callbeschb(xmu,gam1,gam2,gampl,gammi)
  Chebyshev evaluation of \Gamma_1 and \Gamma_2.
  ff=2.d0/PI*fact*(gam1*cosh(e)+gam2*fact2*d)
  f0=e=exp(e)
  p=1.d0/(gampl*PI)
  q=1.d0/(e*PI+gammi)
  pisu2=0. 5d0*pimu
  if (abs(pimu2).lt.EPS) then
    fact3=1.d0
  else
    fact3=sin(pimu2)/pimu2
  endif
  r=P1*pimu2*fact3*fact3
  c=1. d0
  d=-x2*x2
  sum=ff*r*q
  sum1=p
```

Sample page from NUMERICAL RECIPES IN FORTRAN 77: THE ART OF SCIENTIFIC COMPUTING (ISBN 0-521-43064-X) Copyright (C) 1986-1992 by Cambridge University Press. Programs Copyright (C) 1986-1992 by Numerical Recipes Software. Permission is granted to make one paper copy by copyright holders for their personal use or for the personal use of others who have obtained permission. Further reproduction is strictly prohibited. To order Numerical Recipes books or software, visit our website at http://www.nr.com or call 1-800-872-7462 (North America only), or send email to trade@cup.cam.ac.uk (outside North America).
do 13 i=1,MAXIT
    ff=(i*ff+p+q)/(i*i-xmu2)
    c=c*d/i
    p=p/(i-xmu)
    q=q/(i+xmu)
    del=c*(ff+r*q)
    sum=sum+del
    del1=c*p-i*del
    sum1=sum1+del1
    if(abs(del).lt.(1.d0+abs(sum))*EPS)goto 2
  enddo 13
  pause 'bessy series failed to converge'
continue
  rymu=sum
  rjl=sum1*xi2
  rymup=xmu*xi*rymu-ry1
  rjmu=rjmu/(rymup-f*rymu)
else
  a=.25d0-xmu2
  p=-.5d0*xi
  q=1.d0
  br=2.d0*x
  bi=2.d0
  fact=a*xi/(p*p+q*q)
  cr=br+q*fact
  ci=bi+p*fact
  den=br*br+bi*bi
  dr=br/den
  di=bi/den
  dlr=cr*dr-ci*di
  dli=cr*di+ci*dr
  temp=p*dlr-q*dli
  q=p*dli+q*dlr
  p=temp
  do 14 i=2,MAXIT
    a=a+2*(i-1)
    bi=bi+2.d0
    dr=a*dr+br
    di=a*di+bi
    if(abs(dr)+abs(di).lt.FPMIN)dr=FPMIN
    fact=a/(cr*cr+ci*ci)
    cr=br+cr*fact
    ci=bi-ci*fact
    if(abs(cr)+abs(ci).lt.FPMIN)cr=FPMIN
    den=dr*dr+di*di
    dr=dr/den
    di=di/den
    dlr=cr*dr-ci*di
    dli=cr*di+ci*dr
    temp=p*dlr-q*dli
    q=p*dli+q*dlr
    p=temp
    if(abs(dlr-1.d0)+abs(dli).lt.EPS)goto 3
  enddo 14
  pause 'cf2 failed in bessyj'
endif
fact=rjmu/rjl

Equation (6.7.13).
Evaluate CF2 by modified Lentz's method ([5.2]).
6.7 Bessel Functions of Fractional Order

```fortran
rj=rjl1*fact
rjp=rjp1*fact
do i=1,nl
   rytemp=(xmu+i)*xi2*ry1-rymu
   rymu=ry1
   ry1=rytemp
endo
ry=rymu
ryp=xnu*xi*rymu-ry1
return
END
```

```fortran
SUBROUTINE beschb(x,gam1,gam2,gampl,gammi)
INTEGER NUSE1,NUSE2
DOUBLE PRECISION gam1,gam2,gammi,gampl,x
PARAMETER (NUSE1=5,NUSE2=5)
C USES chebev
Evaluates Γ1 and Γ2 by Chebyshev expansion for |x| ≤ 1/2. Also returns 1/Γ(1 + x) and
1/Γ(1 - x). If converting to double precision, set NUSE1 = 7, NUSE2 = 8.
REAL xx,c1(7),c2(8),chebev
SAVE c1,c2
DATA c1/-1.142022680371168d0,6.5165112670737d-3,
* 3.08709017308664d-4,-3.4706269649d-6,6.9437664d-9,
* 3.67795d-11,-1.356d-13/
DATA c2/1.843740587300905d0,-7.68528408447867d-2,
* 1.2719271366546d-3,-4.9717367042d-6,-3.31261198d-8,
* 2.423096d-10,-1.702d-13,-1.49d-15/
x=8.d0*x*x-1.d0
Multiply x by 2 to make range be -1 to 1, and then
gam1=chebev(-1.,1.,c1,NUSE1,xx)
gam2=chebev(-1.,1.,c2,NUSE2,xx)
gampl=gam2-x*gam1
gamm1=gam2+x*gam1
return
END
```

**Modified Bessel Functions**

Steed’s method does not work for modified Bessel functions because in this case CF2 is purely imaginary and we have only three relations among the four functions. Temme [3] has given a normalization condition that provides the fourth relation.

The Wronskian relation is

\[ W = I_{\nu} K'_{\nu} - K_{\nu} I'_{\nu} = -\frac{1}{x} \]  

(6.7.20)

The continued fraction CF1 becomes

\[ f_\nu \equiv \frac{I'_{\nu}}{I_{\nu}} = \frac{\nu}{x} + \frac{1}{2(\nu + 1)/x + \frac{1}{2(\nu + 2)/x + \cdots}} \]  

(6.7.21)

To get CF2 and the normalization condition in a convenient form, consider the sequence of confluent hypergeometric functions

\[ z_n(x) \equiv U(\nu + 1/2 + n, 2\nu + 1, 2x) \]  

(6.7.22)

for fixed \( \nu \). Then

\[ K_{\nu}(x) = \pi^{1/2}(2x)^\nu e^{-x} z_{0}(x) \]  

(6.7.23)

\[ \frac{K_{\nu+1}(x)}{K_{\nu}(x)} = \frac{1}{x} \left[ \nu + \frac{1}{2} + x + \left( \nu^2 - \frac{1}{4} \right) \frac{z_1}{z_0} \right] \]  

(6.7.24)
Equation (6.7.23) is the standard expression for $K_\nu$ in terms of a confluent hypergeometric function, while equation (6.7.24) follows from relations between contiguous confluent hypergeometric functions (equations 13.4.16 and 13.4.18 in Abramowitz and Stegun). Now the functions $z_n$ satisfy the three-term recurrence relation (equation 13.4.15 in Abramowitz and Stegun)

$$z_{n-1}(x) = b_n z_n(x) + a_{n+1} z_{n+1}$$  \hspace{1cm} (6.7.25)

with

$$b_n = 2(n + x)$$

$$a_{n+1} = -[(n + 1/2)^2 - \nu^2]$$  \hspace{1cm} (6.7.26)

Following the steps leading to equation (5.5.18), we get the continued fraction $CF_2$

$$\frac{z_1}{z_0} = \frac{1}{b_1 + \frac{a_2}{b_2 + \ldots}}$$  \hspace{1cm} (6.7.27)

from which (6.7.24) gives $K_{\nu+1}/K_\nu$ and thus $K'/K_\nu$.

Temme’s normalization condition is that

$$\sum_{n=0}^{\infty} C_n z_n = \frac{1}{2x} x^{\nu+1/2}$$  \hspace{1cm} (6.7.28)

where

$$C_n = \frac{(-1)^n \Gamma(\nu + 1/2 + n)}{n! \Gamma(\nu + 1/2 - n)}$$  \hspace{1cm} (6.7.29)

Note that the $C_n$’s can be determined by recursion:

$$C_0 = 1, \quad C_{n+1} = -\frac{a_{n+1}}{n+1} C_n$$  \hspace{1cm} (6.7.30)

We use the condition (6.7.28) by finding

$$S = \sum_{n=1}^{\infty} C_n \frac{z_n}{z_0}$$  \hspace{1cm} (6.7.31)

Then

$$z_0 = \left(\frac{1}{2x}\right)^{\nu+1/2} \frac{1}{1 + S}$$  \hspace{1cm} (6.7.32)

and (6.7.23) gives $K_\nu$.

Thompson and Barnett [4] have given a clever method of doing the sum (6.7.31) simultaneously with the forward evaluation of the continued fraction $CF_2$. Suppose the continued fraction is being evaluated as

$$\frac{z_1}{z_0} = \sum_{n=0}^{\infty} \Delta h_n$$  \hspace{1cm} (6.7.33)

where the increments $\Delta h_n$ are being found by, e.g., Steed’s algorithm or the modified Lentz’s algorithm of §5.2. Then the approximation to $S$ keeping the first $N$ terms can be found as

$$S_N = \sum_{n=1}^{N} Q_n \Delta h_n$$  \hspace{1cm} (6.7.34)

Here

$$Q_n = \sum_{k=1}^{n} C_k q_k$$  \hspace{1cm} (6.7.35)

and $q_k$ is found by recursion from

$$q_{k+1} = (q_{k-1} - b_k q_k)/a_{k+1}$$  \hspace{1cm} (6.7.36)

starting with $q_0 = 0, q_1 = 1$. For the case at hand, approximately three times as many terms are needed to get $S$ to converge as are needed simply for $CF_2$ to converge.
To find $K_\nu$ and $K_{\nu+1}$ for small $x$ we use series analogous to (6.7.14):

$$K_\nu = \sum_{k=0}^{\infty} c_k f_k \quad K_{\nu+1} = \frac{2}{x} \sum_{k=0}^{\infty} c_k h_k$$  \hspace{1cm} (6.7.37)

Here

$$c_k = \frac{(x^2/4)^k}{k!}$$

$$h_k = -k f_k + p_k$$

$$p_k = \frac{p_{k-1}}{k - \nu}$$

$$q_k = \frac{q_{k-1}}{k + \nu}$$

$$f_k = \frac{k f_{k-1} + p_{k-1} + q_{k-1}}{k^2 - \nu^2}$$ \hspace{1cm} (6.7.38)

The initial values for the recurrences are

$$p_0 = \frac{1}{2} \frac{1}{\nu} \Gamma(1 + \nu)$$

$$q_0 = \frac{1}{2} \frac{1}{\nu} \Gamma(1 - \nu)$$

$$f_0 = \frac{\nu \pi}{\sin \nu \pi} \left[ \cosh \sigma \Gamma_1(\nu) + \frac{\sin \sigma}{\sigma} \ln \left( \frac{2}{x} \right) \Gamma_2(\nu) \right]$$ \hspace{1cm} (6.7.39)

Both the series for small $x$, and CF2 and the normalization relation (6.7.28) require $|\nu| \leq 1/2$. In both cases, therefore, we recurse $I_\nu$ down to a value $\nu = \mu$ in this interval, find $K_\mu$ there, and recurse $K_\nu$ back up to the original value of $\nu$.

The routine assumes $\nu \geq 0$. For negative $\nu$ use the reflection formulas

$$\nu = x \nu$$

$$\frac{\nu}{\sin \nu \pi} \left[ \cosh \sigma \Gamma_1(\nu) + \frac{\sin \sigma}{\sigma} \ln \left( \frac{2}{x} \right) \Gamma_2(\nu) \right]$$ \hspace{1cm} (6.7.40)

Note that for large $x$, $I_\nu \sim e^x$, $K_\nu \sim e^{-x}$, and so these functions will overflow or underflow. It is often desirable to be able to compute the scaled quantities $e^{-x} I_\nu$ and $e^x K_\nu$. Simply omitting the factor $e^{-x}$ in equation (6.7.23) will ensure that all four quantities will have the appropriate scaling. If you also want to scale the four quantities for small $x$ when the series in equation (6.7.37) are used, you must multiply each series by $e^x$.

SUBROUTINE bessik(x,xnu,ri,rk,rip,rkp)
INTEGER MAXIT
REAL ri,rip,rk,rkp,x,xnu,XMIN
DOUBLE PRECISION EPS,FPMIN,PI
PARAMETER (EPS=1.e-10,FPMIN=1.e-30,MAXIT=10000,XMIN=2.,
* PI=3.141592653589793d0)
C USES beschb

Returns the modified Bessel functions $r_i = I_\nu$, $r_k = K_\nu$ and their derivatives $r_{ip} = I'_\nu$, $r_{kp} = K'_\nu$, for positive $x$ and for $xnu = \nu \geq 0$. The relative accuracy is within one or two significant digits of EPS. FPMIN is a number close to the machine’s smallest floating-point number. All internal arithmetic is in double precision. To convert the entire routine to double precision, change the REAL declaration above and decrease EPS to $10^{-16}$. Also convert the subroutine beschb.

INTEGER i,l,nl
DOUBLE PRECISION a,a1,b,c,d,del,del1,delh,dels,e,f,fact,
* fact2,ff,gam1,gam2,gammi,gampl,h,p,pimu,q,q1,q2,
* qnew,ril,rill,rimu,rilp,ripl,rilmu,ritt\,s,\,s,\,si,\,si2,\,si2,\,xi,\,xi2,\,xnu,\,xmu2
if(x .le. 0. or.xnu .lt. 0.) pause 'bad arguments in bessik'
Chapter 6. Special Functions

\begin{verbatim}

nl=int(xnu+.5d0)  \hspace{1em} nl is the number of downward recurrences
xnu=xnu-nl     \hspace{1em} of the I's and upward recurrences
xnu2=xnu*xmu    \hspace{1em} of K's. xnu lies between \(-1/2\) and
xi=1.d0/x       \hspace{1em} 1/2.
xi2=2.d0*xi      \hspace{1em} Evaluate CF1 by modified Lentz's method
h=xnu*xi
if (h.lt.FPMIN)h=FPMIN
b=xi2*xnu
d=0.d0

c=h
do : i=1,MAXIT
  b=b+xi2
  d=1.d0/(b+d)
  del=c/d
  h=del*h
  if(abs(del-1.d0).lt.EPS)goto 1
  c=b+1.d0/c
  ril=h*ril
  ripl=ril
  fact=xnu*xi
  do : l=nl,1,-1
    riltemp=fact*ril+ripl
    fact=fact-xi
    ripl=fact*riltemp+ril
    ril=riltemp
  enddo:
  f=ripl/ril
  if(x.lt.XMIN) then
    Use series.
    x2=.5d0*x
    pimu=PI*xmu
    if(abs(pimu).lt.EPS)then
      fact=1.d0
    else
      fact=pimu/sin(pimu)
    endif
    d=-log(x2)
    e=xmu*d
    if(abs(e).lt.EPS)then
      fact2=1.d0
    else
      fact2=sinh(e)/e
    endif
    call beschb(xmu,gam1,gam2,gampl,gammi)
    ff=fact*(gam1*cosh(e)+gam2*fact2*d)
    f=f0.
  else
    Chebyshev evaluation of \(\Gamma_1\) and \(\Gamma_2\).
    ff=fact*(gam1*cosh(e)+gam2*fact2*d)
    sum=ff
    e=exp(e)
    p=0.5d0*e/gampl
    q=0.5d0/(e*gammi)
    c=1.d0
    d=x2*x2
    sum1=p
    do : i=1,MAXIT
      ff=(i*ff+p+q)/(i*i-xmu2)
      c=c*d/i
      p=p/(i-xmu)
      q=q/(i+xmu)
      del=c*ff
      sum=sum+del
      del1=c*(p-i*ff)
  \end{verbatim}
Airy Functions

For positive \( x \), the Airy functions are defined by

\[
Ai(x) = \frac{1}{\pi} \sqrt{x} K_{1/3}(x)
\]  

(6.7.41)
where

\[
Bi(x) = \sqrt{\frac{2}{3}} I_{1/3}(z) + \frac{1}{\pi} K_{1/3}(z)
\]

(6.7.42)

(6.7.43)

By using the reflection formula (6.7.40), we can convert (6.7.42) into the computationally more useful form

\[
Bi(x) = \sqrt{x} \left[ 2 \sqrt{3} I_{1/3}(z) + \frac{1}{\pi} K_{1/3}(z) \right]
\]

(6.7.44)

so that Ai and Bi can be evaluated with a single call to bessik.

The derivatives should not be evaluated by simply differentiating the above expressions because of possible subtraction errors near \(x = 0\). Instead, use the equivalent expressions

\[
Ai'(x) = -\frac{x}{\pi \sqrt{3}} K_{2/3}(z)
\]

\[
Bi'(x) = x \left[ 2 \sqrt{3} I_{2/3}(z) + \frac{1}{\pi} K_{2/3}(z) \right]
\]

(6.7.45)

The corresponding formulas for negative arguments are

\[
Ai(-x) = \sqrt{x} \left[ J_{1/3}(z) - \frac{1}{\sqrt{3}} Y_{1/3}(z) \right]
\]

\[
Bi(-x) = -\frac{x}{\sqrt{3}} \left[ 1 + \frac{1}{\sqrt{3}} J_{1/3}(z) + Y_{1/3}(z) \right]
\]

(6.7.46)

SUBROUTINE airy(x,ai,bi,aip,bip)
REAL ai,aip,bi,bip,x
C USES bessik,bessj

Returns Airy functions \(Ai(x), Bi(x)\), and their derivatives \(Ai'(x), Bi'(x)\).

REAL absx,ri,rip,rj,rjp,rootx,ry,ryp,z,
* PI,THIRD,TWOTH,ONOVRT
PARAMETER (PI=3.1415927,THIRD=1./3.,TWOTH=2.*THIRD,
* ONOVRT=.57735027)

absx=abs(x)
rootx=sqrt(absx)
z=TWOTH*absx*rootx
if(x.gt.0.)then
  call bessik(z,THIRD,ri,rk,rip,rkp)
  ai=rootx*ONOVRT*rk/PI
  bi=rootx*(rk/PI+2.*ONOVRT*ri)
  call bessik(z,TWOTH,ri,rk,rip,rkp)
  aip=-x*ONOVRT*rk/PI
  bip=x*(rk/PI+2.*ONOVRT*ri)
else if(x.lt.0.)then
  call bessjy(z,THIRD,rj,ry,rjp,ryp)
  ai=.5*rootx*(rj-ONOVRT*ry)
  bi=-.5*rootx*(ry+ONOVRT*rj)
  call bessjy(z,TWOTH,rj,ry,rjp,ryp)
  aip=.5*absx*(ONOVRT*ry+rj)
  bip=.5*absx*(ONOVRT*rj-ry)
else
  Case \(x = 0\).
  ai=.35502805
  bi=ai/ONOVRT
6.7 Bessel Functions of Fractional Order

Spherical Bessel Functions

For integer $n$, spherical Bessel functions are defined by

$$j_n(x) = \sqrt{\frac{\pi}{2x}} J_{n+\frac{1}{2}}(x)$$

$$y_n(x) = \sqrt{\frac{\pi}{2x}} Y_{n+\frac{1}{2}}(x)$$

(6.7.47)

They can be evaluated by a call to bessjy, and the derivatives can safely be found from the derivatives of equation (6.7.47).

Note that in the continued fraction CF2 in (6.7.3) just the first term survives for $\nu = 1/2$. Thus one can make a very simple algorithm for spherical Bessel functions along the lines of bessjy by always recursing $j_n$ down to $n = 0$, setting $p$ and $q$ from the first term in CF2, and then recursing $y_n$ up. No special series is required near $x = 0$. However, bessjy is already so efficient that we have not bothered to provide an independent routine for spherical Bessels.

**CITED REFERENCES AND FURTHER READING:**


