The Calculation of Spherical Bessel Functions and Coulomb Functions

A.R. Barnett

Physics Department, University of Manchester, Manchester, M13 9PL, England
and Physics Department, University of Auckland, Auckland, New Zealand

Abstract

An account is given of the Steed algorithm for calculating Coulomb functions and, as a special case, both spherical Bessel and Riccati–Bessel functions. These functions are needed for boundary-condition matching in scattering problems in Atomic and Nuclear physics. Central to the technique is the evaluation of continued fractions and for this calculation Lentz's forward method (modified by Thompson) is recommended. The FORTRAN 77 programs SBESJY, RICBES and COUL90 are presented and described, and some test cases are given. In each program the algorithm returns both the regular and irregular functions as well as their derivatives. The programs are written for real arguments and real orders; references guide the reader to more general codes.

1. Introduction

Coulomb wave functions arise in many problems of physical interest when charged particles scatter from each other. Such scattering is characterised by a relative angular momentum $L\hbar$ (with L a non-negative integer) and by a Sommerfeld parameter $\eta = Z\alpha/\beta$ which gives the strength of the Coulomb interaction. The product of the particle charges is Ze^2, the fine-structure constant $\alpha = e^2/\hbar c(4\pi\epsilon_0)$, and βc is the relative velocity of the particles. With charges of opposite sign, as is frequently the case in Atomic physics, then η is negative, while in Nuclear physics problems generally the charges have the same sign and η is positive. Positron scattering from a nucleus will also have $\eta > 0$. For the scattering of a neutral particle (*e.g.* a neutron) or by a neutral target then the parameter η is zero, resulting in Riccati–Bessel functions in which only the angular momentum effects appear. These functions differ only in a minor way from spherical Bessel functions and both can properly be regarded as special cases of the Coulomb functions.

This review is mainly concerned with the Atomic physics area and thus primarily with the cases where η is not positive. In atomic units the relative energy E, in Rydberg units, is given by $E = E(eV)/13.605\ eV$. Then $E = k^2$ and $\eta = -Z/k$. Typical energies are between $0.5 - 200\ eV$, with corresponding k-values from $(0.2 - 3.8)a_0^{-1}$, and η-values between -5.2 and -0.26. With matching radii of $r = (10 - 200)a_0$ then the dimensionless variable $x = kr$ lies in the range $1 - 1000$. The angular momentum quantum number L will vary from 0 to perhaps 50, while Bessel functions of orders up to 150 may be required. As an example, the parameters for the calculations in Chapter 1 of this book, with energy $54.5\ eV$ are $k = 2.00$, $\eta = -0.50$ (0.50) for electron (positron) scattering off hydrogen ions, and $\eta = 0$ for either particle scattering from neutral hydrogen.

The second-order differential equation satisfied by the Coulomb functions is,

$$w''(x) + [1 - 2\eta/x - L(L+1)/x^2]\, w(x) = 0 \tag{1}$$

(where the primes indicate differentiation with respect to x) and it has two linearly independent solutions. They are chosen to be the regular solution $F_L(\eta, x)$ which is zero at $x = 0$, and the irregular solution $G_L(\eta, x)$ which is infinite at $x = 0$. See also refs[1,2,3,9,10] for alternative forms of (1).

The 'turning point' for the L^{th} partial wave occurs at a value of x

$$x_L = \eta + (\eta^2 + L^2 + L)^{1/2} \tag{2}$$

which is where the bracket [...] is zero in (1); it is also the first point of inflexion for the functions. For negative η, say $\eta = -h$, the $L = 0$ turning point is at the origin, $x_0 = 0$, and for other L-values x_L is always less than L; it lies between $L - h$ (for small h) and $\frac{1}{2}L(L/h)$ (for small L). It will transpire that the computational task is easier when $x > x_L$, so that functions with negative η are obtained more straightforwardly than when η is positive.

For values of x which are greater than x_L the functions take on an oscillatory character, although the 'period' slowly changes. Examples of the functions are shown in Fig.1. For $\eta > 0$ the regular function magnitude is greater than unity, and it slowly decreases towards unity as x grows larger. When $\eta < 0$ the magnitude of $F_L(\eta, x)$ is less than unity and it increases steadily for larger x. In the asymptotic region as $x \to \infty$ they become circular with $F \to \sin\theta_L$ and $G \to \cos\theta_L$, where θ_L is the asymptotic phase given by

$$\theta_L = x \; - \; \eta ln(2x) \; - \; {}^1\!/_2 L\pi \; + \; \arg\Gamma(L+1+i\eta) \tag{3}$$

and in the case when $\eta = 0$ this phase becomes linear in x:

$$\theta_L = x \; - \; {}^1\!/_2 L\pi$$

An additional phase of ${}^1\!/_2\pi$ will be required for spherical Bessel functions because of a different definition which is given in eqn(8).

This article deals with solutions to the Coulomb scattering problem in which the particles have a positive relative energy. Curtis[1] published a detailed discussion for $\eta < 0$ in 1964 for $L = 0, 1, 2$ and for functions closely related to F and G, for use in electron scattering calculations. More recent work for electron scattering is that of Seaton[2] and of Bell and Scott[3]; Seaton dealing with positive, negative and zero energies, while Bell and Scott treated negative energies. The work of Curtis for positive energy was verified numerically by Barnett[4] in 1974. Bardin *et al.*[5] published a comprehensive suite of programs in 1972 containing codes for all real η and x. All of these methods involve some appropriate series expansion, either in powers of x, or asymptotically in powers of x^{-1}. Both F and G are calculated separately, as are their x-derivatives. A new approach to the calculation of Coulomb functions was developed by Steed in the late 1960s and was published by Barnett *et al.*[6] in 1974. The functions and their derivatives are calculated *together* in an interdependent way, and two continued fractions are used (§6). Virtually all previous work, especially that of Bardin *et al.,*[5] was verified and in some cases extended in this paper. A detailed description of this algorithm and a careful comparison with other approaches was given by Barnett[7] in 1982. The original program[6] RCWFN was superseded by a more comprehensive version,[4] called COULFG, which includes the calculation of both Bessel and spherical Bessel functions (of the first and second kind *i.e.* $j_n(x), y_n(x)$) as well as the Coulomb functions. The version presented here in §7, COUL90, has been further improved in one important respect which, however, does not affect the algorithm, the range of the calculations or the results already given, both in ref.[4] and the earlier work. Comments on the methods of COULFG have been made by Nesbet[8] with suggestions for improvements for both Coulomb–function calculations when $x < x_{L=0}$, and especially for spherical Bessel functions.

Solutions for negative energy (including the bound states) are given by Whittaker functions and, for both $\eta < 0$ and $\eta > 0$, can be obtained by using the recent program COULN of Noble and Thompson[9] which is well suited to electron scattering.

The algorithm of COULFG is not restricted to integer L-values, as refs[4,10] demonstrate, and it can hence be used for scattering solutions to relativistic problems for which the Klein–Gordon equation or the Dirac equation are appropriate. Here an equivalent non-integer L is required; for small L-values this can be imaginary. Extensions of the concepts underlying the calculations to *complex arguments* are presented in the program COULCC, which was published in 1985 by Thompson and Barnett.[11] The range of each of the three variables has been extended into the complex plane. The program has been recently been incorporated into the IMSL SFUN library. A description has been given in ref.[12] of the various approaches required in the different parameter regions, with references to earlier and more restricted work. The code COULCC also computes Bessel functions with complex arguments and order; however, for real order, BESSCC for modified Bessel functions is a more efficient code,[11] which incorporates similar principles.

2. Spherical Bessel Functions

Spherical Bessel functions, and their close relatives the Riccati–Bessel functions, are required frequently in Atomic physics calculations and in many other branches of Physics, for example in the calculations of Mie scattering in Optics. In Chapter 1 of this book they emerge as the asymptotic wavefunctions for the problem of charged particles scattering from neutral atoms or molecules. Similarly, they are required for the description of the scattering of neutrons from nuclei. Despite this they rather tend to be the poor relations of the numerical analysis world. They are covered in the encylopaedic *Handbook* of Abramowitz and Stegun[13] but they are not mentioned, for example, in the *Numerical Recipes*,[14] chapter 6 on Special Functions, where Bessel functions (*i.e.* with *cylindrical* symmetry) are treated, and few, if any, large-scale libraries (such as NAG, IMSL *etc.*) have suitable subroutines. The reason is simply that all orders are expressible as sums of polynomials in x^{-1} multiplied by $\sin x$ and $(-\cos x)$, and recurrence relations connect consecutive orders. The computational task appears trivial. Nevertheless there are difficulties, shared with the evaluation of more general Bessel functions and Coulomb functions, which stem directly from the relevant differential equation. A useful discussion, intended for the physicist exploring numerical analysis, of some of these difficulties in the case of cylindrical Bessel functions, is that of Koonin[15](§4.1), which also contains the details of an fixed-accuracy program to evalute them. Similarly, *Numerical Recipes* by Press *et al.*[14] contains an excellent coverage of the counter-intuitive aspects of the evaluation of special functions, of difference and of differential equations, as well as numerous programs.

Very recently however, Press and Teukolsky, two of the authors of *Numerical Recipes*, have published in their regular column[34] programs for $J_\nu(x)$, $Y_\nu(x)$ for real cylindrical Bessel functions and for the modified $I_\nu(x)$, $K_\nu(x)$ Bessels

of real argument x and real order ν. These programs directly adopt Steed's method and the extensions and the techniques given in the complex-argument, real-order code BESSCC[11] of Thompson and Barnett. The Press-Teukolsky codes are thus more suitable for real arguments and are faster since they are less general. As examples of their use, calling programs are provided to find Airy functions and spherical Bessel functions. This route makes the spherical Bessels conceptually a special case of the usual Bessel function; it is also evident[23] that *both* are special cases of the Coulomb functions. No test cases are given, nor are ranges specified on the parameters, but in general the same conclusions as those presented in this article will apply.

Two programs have been recently published for the specific calculation of spherical Bessel functions, by Gillman and Fiebig[16] and by Lentz.[26] They are discussed at the end of §5.

For **spherical Bessel functions**, with solutions $j_n(x)$ and $y_n(x)$, the differential equation is:

$$w'' + 2w'/x + [1 - n(n+1)/x^2]\, w = 0 \tag{4}$$

In this equation (see ref.[13] eqn 10.1.1) x is a real variable while n is an integer which is positive, negative or zero and which will be identified with the non-negative angular momentum. Many of the properties of j_n and y_n follow directly from their identification as Coulomb functions with $\eta = 0$, as is done below through eqn(11). The results are

$$j_n(x) = x^{-1} F_n(0,x) \qquad\qquad y_n(x) = -x^{-1} G_n(0,x) \tag{5}$$

¿From this and the properties of F and G we immediately deduce that j and y will display an oscillatory nature for x-values larger than $\sqrt{(n^2+n)}$, their magnitude will decrease as x^{-1} for these larger x-values, and that y will diverge towards $-\infty$ as x approaches zero. The first two orders of the **spherical Bessel functions** are:

$$\begin{aligned} j_0(x) &= x^{-1}\sin x & \qquad y_0(x) &= -x^{-1}\cos x \\ j_1(x) &= x^{-2}\sin x - x^{-1}\cos x & \qquad y_1(x) &= -x^{-2}\cos x - x^{-1}\sin x \end{aligned} \tag{6}$$

and recurrence relations (§3) link sets of three consecutive orders. The derivatives $j_0'(x),\ y_0'(x)$ follow from (6).

The two solutions j_n and y_n are called the *regular solution* and the *irregular solution* respectively. This describes their behaviour at the origin of x where the irregular solution diverges to $-\infty$ as $-(2n-1)!!x^{-(n+1)}$ and the regular one goes to zero as $x^n/(2n+1)!!$, (and with $j_0(0) = 1$). The irregular solution, $y_n(x)$, is called the spherical Neumann function and occasionally given the symbol $n_n(x)$. The following linear combinations are the spherical Hankel functions,

$$\begin{aligned} h_n^{(1)}(x) &= j_n(x) + iy_n(x) \\ h_n^{(2)}(x) &= j_n(x) - iy_n(x) \end{aligned} \tag{7}$$

As x becomes large the equations of ref.[13] (9.2.1, 9.2.17 and those following 10.1.26) show that

$$\begin{aligned} j_n(x) &\to x^{-1}\cos(x - {}^1\!/_2 n\pi - {}^1\!/_2\pi) \;=\; x^{-1}\sin(\theta_n - {}^1\!/_2\pi), \text{ and} \\ y_n(x) &\to x^{-1}\sin(x - {}^1\!/_2 n\pi - {}^1\!/_2\pi) \;=\; -x^{-1}\cos(\theta_n - {}^1\!/_2\pi) \end{aligned} \tag{8}$$

This difference in phase definition (the additional ${}^1\!/_2\pi$ which interchanges the role of sine and cosine in (8)) is carried over to the equivalents of the Hankel functions for Coulomb scattering. For this Coulomb case the expression for the outgoing wave is chosen to be

$$H_L^+(\eta, x) \;=\; G_L(\eta, x) \;+\; iF_L(\eta, x) \;\to\; \exp(i\theta_L) \text{ as } \quad x \to \infty \tag{9}$$

and similarly for H^- the incoming wave. For $\eta = 0$ the relations are

$$h^{(1)}(x) = -(i/x)H^+(0, x) \qquad\qquad h^{(2)}(x) = (i/x)H^-(0, x) \tag{10}$$

with the ${}^1\!/_2\pi$ phase difference remaining. The $H^\pm$ scattering functions must not be confused with the Hankel functions for cylindrical Bessels, $H_n^{(1)}$ and $H_n^{(2)}$ (see ref.[13] eqn 9.1.3, 9.1.4, 9.1.6), which are analogous to (7).

To obtain the **Riccati–Bessel functions** we transform (4), by removing the first-derivative term, into

$$w'' \;+\; [1 - n(n+1)/x^2]\, w = 0 \tag{11}$$

whose solutions (ref.[13] §10.3) are $xj_n(x)$ and $xy_n(x)$. No special symbol has been given to them in the mathematical literature although, for complex argument z, a consistent notation is used[17,18,19] for Mie scattering:

$$\begin{aligned} \psi_n(z) &\;=\; zj_n(z) \\ \chi_n(z) &\;=\; -zy_n(z) \\ \zeta_n(z) &\;=\; \psi_n(z) \;+\; i\chi_n(z) \end{aligned} \tag{12}$$

The Riccati–Bessel properties are briefly treated in the *Handbook*[13][§10.3]; here, however, $\chi_n \;=\; +zy_n$. For orders $n \;=\; 0, 1$ we then have, adopting ref.[13]

$$\begin{aligned} xj_n(x) &= \sin x & xy_0(x) &= -\cos x \\ xj_1(x) &= x^{-1}\sin x \;-\; \cos x & xy_1(x) &= -x^{-1}\cos x \;-\; \sin x \end{aligned} \tag{13}$$

On comparison of (11) with (1) it is evident that the Riccati–Bessel functions are the same as the Coulomb functions with $\eta = 0$, while (8) and (9) show that the sign of the irregular function chi_n is reversed relative to $G_L(0, x)$. The Riccati–Bessel functions thus behave like $\sin\theta_n$ and $(-\cos\theta_n)$ as x becomes large, where the asymptotic phase, θ_n, is given by (3). Alternatively this can be written $\cos(\theta_n - {}^1\!/_2\pi)$, $\sin(\theta_n - {}^1\!/_2\pi)$ as in (8). The subroutine RICBES retains the *Handbook* definitions in which the Riccati–Bessel is the argument x times the corresponding spherical Bessel function.

3. Recurrence Relations for Spherical Bessel Functions

Each of the four spherical Bessel functions $j_n,\ y_n,\ h_n^{(1)},\ h_n^{(2)}$ obeys recurrence relations [ref.[13] 10.1.19 – 10.1.22] which connect the functions of order $(n-1)$, n and $(n+1)$. Using g_n to represent any of these four functions the relations are:

$$g_{n-1} \ - \ \frac{2n+1}{x}\, g_n \ + \ g_{n+1} = 0 \tag{14}$$

and

$$n g_{n-1} \ - \ \frac{2n+1}{x}\, g'_n \ - \ (n+1) g_{n+1} = 0. \tag{15}$$

These can be rewritten in a form which is suitable for **downward recurrence**, connecting two successive orders and a derivative,

$$g_{n-1} = S_{n+1} g_n \ + \ g'_n \tag{16}$$

$$g'_{n-1} = S_{n-1} g_{n-1} \ - \ g_n \tag{17}$$

in which $S_n = n/x$. The equivalent expressions for **upward recurrence** are equations (17) and (16) rearranged:

$$g_{n+1} = S_n g_n \ - \ g'_n \tag{18}$$

and

$$g'_{n+1} = g_n \ - \ S_{n+2} g_{n+1} \tag{19}$$

The recurrence relations (14) – (17) are an alternative way of expressing the differential equation (4) as difference equations.

It is well known (*e.g.* ref.[14] §5.4, ref.[15] §4.1, and ref.[8] §3) that a recurrence relation is numerically unstable in the direction in which the function is decreasing. Successive values are computed as small differences between nearly equal terms and all accuracy is soon lost. This occurs for the regular function $j_L(x)$ once $L > x$ for any fixed value of x: $j_L(x)$ *decreases monotonically* as a function of L , so that upward recurrence in n of $j_n(x)$ is unstable. Conversely, since the irregular function $y_L(x)$ *increases* as L increases, once $L > x$, upward recurrence is stable. Thus for $L > x$ we must use *downward recurrence in* n to calculate the values of $j_n(x)$ $(n = L, L-1, ..., 3, 2, 1, 0)$; similarly *upward recurrence* must be used for $y_n(x)$. In the region where $L < x$ then the functions have an oscillatory character and recurrence in both directions is stable.

Hence, to compute both $j_n(x)$ and $j'_n(x)$ for all orders from 0–L by using (16) and (17) we need $j_L(x)$ and $j'_L(x)$, or their ratio, for the maximum $n = L$ and to find the smaller n-values by downward recurrence, normalising at $j_0(x)$ with (6). On the other hand, starting with $y_0(x)$ and $y'_0(x)$ from (6), the upward recurrence equations (18), (19) will yield stable values for the irregular functions and their derivatives for each value of the order.

¿From (18), which is satisfied by $j_n(x)$, we see that the logarithmic derivative is given by

$$\frac{j'_n}{j_n} = S_n \; - \; \frac{j_{n+1}}{j_n} \tag{20}$$

and from (14) it is easy to derive a continued fraction for the ratio of successive orders:

$$\frac{j_{n+1}}{j_n} \; = \; \frac{1}{(2n+3)/x -} \; \frac{1}{(2n+5)/x -} \; \frac{1}{(2n+7)/x \; - ...} \tag{21}$$

The coefficents in the denominators are $S_k + S_{k+1} \equiv T_k$ for k starting at $n+1$. The equation is implicit in [10, 9.1.73 and 10.1.1]; it was derived (for Coulomb functions) as eqn(2.19) of ref.[7] and by a different method in ref.[6] (It was quoted erroneously in ref.[23] in equations (37), (38) and (39) – in each case the first term should be dropped.) A most important point is that (21) applies *only to the regular solution.* It is not apparent from the derivation that the formula *does not hold* for the irregular functions y_n or h_n, *even though they, too, satisfy all the recurrence relations.* The explanation of this remarkable result, and an indication of its generality, appears in Gautschi[24] and in refs[20,22]: only the minimal (*i.e. regular*) solution has the property (21). Also, combining (20) and (21) we find,

$$f \equiv \frac{j'_n}{j_n} \; = \; S_n \; - \; \frac{1}{T_{n+1} -} \; \frac{1}{T_{n+2} -} \; ... \; \frac{1}{T_k \; - ...} \tag{22}$$

The numerical evaluation of the continued fraction f uses the fact that eventually the denominator $T_k = (2k+1)/x$ will become large enough so that the value of f can be found by terminating (22) at the k^{th} step while retaining a chosen accuracy.

The problem of calculating the spherical Bessel functions becomes that of computing the continued fraction f to sufficient accuracy. Equation (22) will be referred to as CF1, whereas CF2 will refer to a second continued fraction: $p \; + \; iq \; = \; H'/H$ which is discussed in §6, eqn(34). It may be noted that the reciprocal of (21) is also used, *e.g.* by Lentz,[17] and that it is a more complicated expression.

4. Evaluation of the Continued Fraction

Continued fractions are mentioned in Abramowitz and Stegun[13] in §3.10 and are covered in Numerical Recipes[14] in chapter 5.2 . There is an intimate relation between recurrence relations of the type (14) and continued fractions: a full discussion and literature guide appears in Chapter 4 of van der Laan and Temme,[20] particularly §4.8, and much valuable and detailed information appears in Wynn[21] and in Wimp.[22] The forward evaluation suggested in refs[13,14] is not to be recommended: a backward recurrence algorithm originally due to Miller (see ref.[14] §5.4 and §6.4, ref.[20] §3.3, or ref.[15] Chapters 4 and 5) is superior and is very widely used in the improved form given by Gautschi.[24]

Steed's method[6] for continued fractions is a stable forward recurrence and it has further advantages (discussed in ref.[23] §4). It was adopted in the author's earlier programs.[4,6,10,23] This method involves a *summation* when updating f_{n-1} to become f_n and was hence subject to rare numerical cancellation errors when a certain denominator approached zero [ref.[10] eqn(11), refs[8,25,26]]. Thus Steed's method for continued fractions does not compute the result every where to uniform accuracy. In cases where there are no zeros involved (*e.g.* in calculating CF2 rather than CF1, see end of §3) then there can be no objection in principle to using it (as does BESSCC and part of the COULCC code). The same method, although unnamed, has been used by Gautschi and Slavik[27] and is also described and recommended in the classic Gautschi paper.[24] Subsequent authors have not been alert to the manifest advantages of Steed's method. Thompson and Barnett[11] in 1987 released the code BESSCC, which treats complex-argument Bessel functions and evaluates the complex continued fraction involved (CF2) by Steed's method.

The method of choice for continued fractions, however, is none of the above but the **method of Lentz**[17] together with the 'zero shifts' of both numerator and denominator discussed by Jaaskelainen and Ruuskanen.[25] That paper, however, proposed an elaborate change to to the algorithm, whereas Thompson shows in Appendix III of ref.[12] how to achieve these shifts with minimum change to the Lentz method. Explicitly, the Lentz–Thompson algorithm reads:

***L–T* algorithm for the forward evaluation of continued fractions**
Given the n^{th} convergent of a CF, *i.e.* the sum to n terms of

$$f_n = b_0 + \frac{a_1}{b_1+}\,\frac{a_2}{b_2+}\cdots\frac{a_{n-1}}{b_{n-1}+}\,\frac{a_n}{b_n} \tag{23}$$

then evaluate $f = lim_{n\to\infty}(f_n)$ to an accuracy *acc* with the algorithm:

$f_0 := b_0; \quad if\ (f_0 = 0)\ f_0 = small$
$C_0 := f_0, \quad D_0 := 0$
$for\ n = 1, limit \quad do\ begin$
$\quad C_n := b_n + a_n\ /\ C_{n-1}; \quad if\ (C_n = 0)\ C_n = small$
$\quad D_n := b_n + a_n \times D_{n-1}; \quad if\ (D_n = 0)\ D_n = small$
$\quad D_n := 1/D_n$
$\quad \Delta_n := C_n \times D_n; \quad f_n := f_{n-1}\Delta_n$
$\quad if\ (\mid \Delta_n - 1 \mid < acc)\ exit$
end

Notes:

1. If at any stage we represent f_n as A_n/B_n(before cancellation), then the two quotients which are used in the *L–T* algorithm are $C_n = A_n/A_{n-1}$ and $D_n = B_{n-1}/B_n$.
2. The parameter *small* should be some non-zero number less than typical values of the quantity $acc\times \mid b_n \mid$ *e.g.* 10^{-50} for typical double precision calculations in which a sensible choice of *acc* is 10^{-14}.

3. The zero tests are to a working accuracy, say *tol*, which can be chosen to avoid divide-by-zero error checks. It could be taken as equal to *small*. Thus algebraic conditions such as '$if\,(D_n = 0)$' are to be read as computation conditions: '$if\,(abs(D_n) < tol)$'.
4. The constant *limit* is an integer designed to abort the loop if necessary. (Our programs use $limit = 20,000$. For small L-values the algorithm would require this number of iterations for an x-value of about 20,000. Graphs, estimates and explanations for this behavior are given in ref.[7])
5. All the methods above, and others (ref.[12] §3.3), apply for *complex* values of a_k, b_k.

Lentz's method was developed in 1975 to deal with complex arguments in calculations of Mie scattering. Lentz required the Riccati–Bessel logarithmic derivative $[zj_n(z)]'/[zj_n(z)]$ for a range of values of n and $z = x + iy$. More extreme parameter values were treated by Wiscombe[18] in 1980. It must be realised that when z lies away from the real axis, the natural directions for recurrence in order may change, and considerably more care needs to be exercised in the computations. Examples of difficulties are given in Lentz[17,26] and in ref.[12] §3.1; the effect complicates the coding[11] of COULCC by forcing the monitoring of the moduli of the functions during recurrence.

The Lentz–Thompson method given above is quite general and it seems to retain the advantages of Steed's method and to remove the (minor) problems. In Lentz's papers[17,26] he advocates a slightly different error-correcting procedure. The choice is probably a matter of taste. However it should be restated that nothing is particularly critical in the selection of the number *small* (see 2 above.) In addition, the Lentz–Thompson method can also be recommended for complex arguments (where these numerical cancellation problems are even less likely). It is not at all clear why ref.[26] discards it for this use.

In the BESSCC paper[11] Thompson gave a listing of a complex-argument spherical Bessel code SBESJH which uses the new *L–T* continued fraction algorithm. (The range of usefulness of the program in fact is greater than that assumed by Thompson, and see also Ross.[19]) The code SBESJY in the next section also adopts this technique.

5. The Programs SBESJY and RICBES

A listing of the spherical Bessel program SBESJY, for real argument, appears in Fig.2. The four functions $j_L(x),\ j'_L(x),\ y_L(x),\ y'_L(x)$ are returned for all orders from 0 to $Lmax$. It is an extension of the program listing of the same name which was published by Barnett[23] in 1981, and while the principles are identical, the realisation is rather different. FORTRAN 77 coding is used, the arrays are indexed from 0 to $maxL$, the *L–T* algorithm replaces the Steed algorithm for the calculation of f *i.e.* CF1 (which, as programmed in ref.,[23]

may fail for isolated values of x, *e.g.* for $x = \sqrt{15}$), the cosine and sine functions are used to find the $L = 0$ solution rather than the square root function, the results for $x = 0$ are given, and all four functions are computed.

The closely related program RICBES calculates the Riccati–Bessel functions of eqn(12), $\psi_L(x)$, $\chi_L(x)$ and their x-derivatives $\psi'_L(x)$, $\chi'_L(x)$. These functions are equally $F_L(0,x)$, $-G_L(0,x)$, $F'_L(0,x)$, $-G'_L(0,x)$ and so can also be obtained as a special case of COULFG or COUL90 (see §7).

In the program SBESJY both the two recurrences and the continued fraction are evaluated as Bessel functions. First CF1 = f is calculated from (22) with $n = Lmax$, $a_k = -1$ and $b_k = T_k$. This yields a value $j'_{Lmax} = f j_{Lmax}$ when the unnormalised j_{Lmax} is taken as unity. Then, if $Lmax > 0$, the *do loop* to label 2 implements the downward recurrence (16) and (17), until unnormalised values for j_0, j'_0 are found. Relative values of j_L and j'_L for each L from this procedure are stored ready for normalisation, which is accomplished by (6). The values of y_0, y'_0 are also obtained from (6) and the recurrence relations (18), (19) used to find the remaining L-values from 1 to $Lmax$. It is clear that if a program without derivatives were required then eqns(16), (17) could be replaced by (15), with (21) used instead of (22, CF1). SBESJY was tested against SBESJ,[23] COULFG,[4] the values in ref.[13] and the two programs described below, DPHRIC[16] and cfbessel[26]. The range of parameters was $x = 0.01 - 1000.0$ and $L = 0 - 1000$, and in general, all results agreed to better than a relative accuracy of 10^{-12} when the *accur* parameter is set to 10^{-15}. Some test data are given in §8. The first two programs are unusual in *not* using the functions $\cos(x), \sin(x)$ in the calculation. Those which do may suffer from truncation error when a large value of x is reduced to the range $\pm 1/2\pi$ by subtracting $N\pi$, where N is a suitable integer. This reduction will be compiler dependent and can be programmed, to some extent, by methods described in Cody and Waite,[33] Chapter 8, p 136. The value π is written as C1 + C2, where C1 = 3217/1024 = 3.1416 01562 50000 is exactly machine representable and C2 = –8.9089 10206 76153 73566 E–6.

For the program RICBES both the two recurrences and the continued fraction are now evaluated as Coulomb functions. First CF1 = f is calculated from (33) with $\eta = 0$ (and hence $R_k^2 = 1$ and $T_k = (2k+1)/x$), and with $n = Lmax$. It differs from (22) only by $1/x$. This yields a value $F'_{Lmax} = f F_{Lmax}$ when the unnormalised value of F_{Lmax} is taken as unity. Then, if $Lmax > 0$, the *do loop* to label 2 implements the downward recurrence (26) and (27) until unnormalised values for F_0, F'_0 *i.e.* ψ_0, ψ'_0, are found. Relative values of all the F_L and F'_L are stored ready for normalisation, which is accomplished by (13). The values of $\chi_0 = -G_0$, $\chi'_0 = -G'_0$ are also obtained from (13) and the upward recurrences (29), (30) used to find the remaining L-values.

A recent program to compute $j_L(x)$ and $y_L(x)$ –called $n_L(x)$– is that of Gillman and Fiebig.[16] Constructive criticisms of its archaic and awkward style have been given by Welch[28] who has presented rewritten versions in order to emphasise that FORTRAN 77 and FORTRAN 90 programmers have no excuse for producing non-structured programs. The method of Gillman and Fiebig is to calculate modified functions $u_L(x)$, $v_L(x)$ in place of $j_L(x)$, $y_L(x)$ from which

factors of $x^L/(2L+1)!!$ and $-(2L-1)!!/x^{L+1}$ have been extracted. These scaled functions tend to unity as $x \to 0$ (*cf* §2 between eqns(6) and (7).) A version of Miller's method is used to obtain the relative values of u_L for all the L-values considered (0–1000), with x in the range 0.01–100.0. The values are normalised by the Wronskian relation for the $u,\ v$ functions.

The reason for choosing $u_L(x)$ and $v_L(x)$ is that computational overflow and underflow conditions are removed. In the days when representable real numbers were restricted $10^{\pm 38}$ or $10^{\pm 70}$ then this was indeed a problem. It is commonplace for modern FORTRAN 77 compilers to offer $10^{\pm 308}$ in double precision variables (as does the Lahey F77L compiler, whose version 4.0 was used in this work) which is large enough to remove the need for special programming in normal physical applications.

A second point is that the u, v recurrences are *assumed* to be stable in the same directions as are the j, y recurrences. This is not an obvious fact (*e.g.* Lentz[17,26]) but the stability was proved rigorously in 1980 by O'Brien[29] for real arguments. He claimed, further, that no Miller's method is necessary, replacing it by two evalutions of 'a rapidly convergent series'. (No numerical details were given and O'Brien's results have not yet appeared as a program.)

The third point is that tests for overall accuracy in ref.[16] only address numerical consistency, and not the method. This is recognised by Gillman and Fiebig who say that 'it is of some value'. It is rather puzzling to a user who sees tests aimed at a relative accuracy of 2.10^{-8} apparently produce results to better than 10^{-15}, which is about machine accuracy. One reason may be the tests measure the *rms* Wronskian deviation from unity, which reduces by the square root of the number of L-values, and here by a factor of 30. But the *same Wronskian* is used to normalise the u_L-values, so the tests are not independent. There is no need for this choice of normalisation for the exact value of $u_0(x)$ is $\sin(x)/x$ and should be chosen instead. However, the choice of this correct normalisation gives exactly the same result to machine precision. The correct answer involves a more subtle point. By choosing to compute the rms deviation *over all L-values* Gillman and Fiebig minimise the fact that the downwards recurrence of Miller's method is most at error *at the high L-values* which is worst when x/L is largest. An independent calculation[26] shows that $j_{1000}(x)$ is in error by a fraction 10^{-3} for $x = 100$, which falls to 10^{-7} for $x = 0.5$. The healing step in L is small: even for $L = 999$ the errors improve to 10^{-5} and 10^{-15}. The last few L-values should not be used, and it would seem that the choice of 1000 for the starting L is too large and their estimate of an error $\epsilon \simeq (x/2L)^6$) is optimistic for L itself. Making the correct choice of a starting L is the hard part of Miller's method. Of course these comments relate only to the regular solution $j_n(x)$; for the $y_n(x)$-values depend only on their (accurate) starting values $y_0(x),\ y_0'(x)$ since the upward recurrence is stable. The value of $j_{1000}(100)$ is $5.32338\ 16172 \times 10^{-872}$, while $j_{1000}(0.5) = 6.06344\ 55462 \times 10^{-3172}$, and are clearly of mathematical rather than of physical interest.

It should be noted that two of the tables in the Gillman and Fiebig article contain sign errors, which do not occur when the published program is run. Specifically, in Table III, for $j_L(100)$ the entries for $L = 2, 3, 4, 6, 11, 12$ and 16

have the wrong sign, as do the $y_L(100)$ entries for $L = 19$ and $L = 50$. In Table V, $v_{30}(100)$ also has the incorrect sign.

The approach of ref.[16] in factorising out the small-x behaviour, when reinforced by theory,[29] works well for spherical Bessel functions. The underflows and overflows in reapplying the normalisation to recover j_L, y_L can easily be trapped by extracting, say, powers of $10^{\pm 200}$ when appropriate, as is done by Lentz[26]. There is nothing to prevent the CF1 calculation in SBESJY replacing Miller's method to remove the inaccuracy near the maximum L-value.

The program *cfbessel.for*, published by Lentz,[26] calculates a single value of $j_n(z)$, for complex z. His algorithm for the forward evaluation of continued fractions creates an infinite product which is terminated when a given accuracy is reached. It is a flexible and elegant concept, applicable to a wide range of complex arguments, although just what the limits are is not discussed by Lentz. Despite the different appearances it is the same algorithm which is described in §4. As given, the Lentz program only results in $j_n(z)$ and not $y_n(z)$ or the derivatives, and intermediate n-values are not calculated. Thompson's program SBESJH given in ref.[11] includes all these features. These two codes, together with the code *sphbes* of Press and Teukolsky,[34] will be compared in detail in a subsequent publication.

6. Recurrence Relations for Coulomb Functions

Each of the four Coulomb functions F_n, G_n, H_n^+, H_n^- (see eqn(19) obeys recurrence relations [ref.[13] 14.2.3, 14.2.1 and 14.2.2] which connect the functions of order $(n-1)$, n and $(n+1)$, thus

$$R_n w_{n-1} - T_n w_n + R_{n+1} w_{n+1} = 0 \tag{25}$$

$$w_{n-1} = [S_n w_n + w'_n]/R_n \tag{26}$$

$$w'_{n-1} = S_n w_{n-1} - R_n w_n \tag{27}$$

The coefficients are:

$$\begin{aligned} R_k &= \sqrt{1+\eta^2/k^2} \\ S_k &= k/x + \eta/k \\ T_k &= S_k + S_{k+1} = (2k+1)[x^{-1} + \frac{\eta}{k^2+k}] \end{aligned} \tag{28}$$

Equations (26) and (27) connecting two successive orders and a derivative are in a form suitable for **downward recurrence** in n. The equivalent expressions for **upward recurrence** in n are rearrangements of (27) and (26),

$$w_{n+1} = [S_{n+1} w_n - w'_n]/R_{n+1} \tag{29}$$

and

$$w'_{n+1} = R_{n+1}w_n \; - \; S_{n+1}w_{n+1} \tag{30}$$

The recurrence relations (25), (26), (27) are an alternative way of expressing the differential equation (4) as a difference equation. As Fröberg[30] shows, not even two of these recurrence relations are independent. The equation analogous to the spherical Bessel (15) loses its simplicity and becomes

$$S_{n+1}R_nw_{n-1} \; - \; T_nw'_n \; - \; S_nR_{n+1}w_{n+1} = 0,$$

and this is not particularly useful.

The boundary between solutions of oscillating and monotonic character occurs at the turning point x_L of (2), when L is fixed, or alternatively, for fixed x it is found at L_{TP}. This is also given by (2) and equivalently by

$$x^2(R_L^2 - S_L^2) \; = \; 1$$

or

$$L_{TP} \; = \; (x^2 - 2\eta x + {}^1\!/_4)^{1/2} - {}^1\!/_2$$

Since $F_L(x)$ decreases to zero as L increases beyond L_{TP}, the stable direction (the function must not decrease) is *downward recurrence in* n to calculate the values of $F_n(x)$ $(n = L, L-1, ..., 3, 2, 1, 0)$. Similarly *upward recurrence* must be used for $G_n(x)$. In the region where $L < L_{TP}$ all the functions have an oscillatory character and recurrence in both directions is stable.

Hence to compute both $F_n(x)$ and $F'_n(x)$ for all orders from 0–L it appears that we need both functions for the maximum order L and to use (26), (27) to find lower n-values, as was the case for the spherical Bessels in §3. Now, however, there is no easy way to normalise at $n = 0$. Methods of finding $F_0(\eta, x)$ and $F'_0(\eta, x)$ directly are given[5] in the programs of Bardin *et al.* They demand detailed numerical analysis and a knowledge of most of the properties of Coulomb functions near the origin, in order to deal with the full range of x and η. Similarly intricate study is required for the separate evaluation of $G_0(\eta, x)$ and $G'_0(\eta, x)$, as is also shown by Strecock and Gregory,[31] so that the upward recurrence equations (29), (30) will yield the irregular functions and their derivatives for each value of the order. In all, ten separate subroutines for the different methods are required. Bardin *et al.* check their independent evaluation of the F and G functions, and their derivatives, by computing the value of the Wronskian, which is unity for Coulomb functions:

$$F'_L(\eta, x)G_L(\eta, x) \; - \; F_L(\eta, x)G'_L(\eta, x) \; = \; 1 \tag{31}$$

although this test is not foolproof (as was shown in ref.[6] section 5.)

Steed's algorithm for calculating Coulomb functions (and hence[23,4,10] Bessel functions, spherical Bessel and Riccati–Bessel functions, Airy functions *etc.*) is based on a different approach, which has the significant merit that *no detailed information about the function behaviour at the origin is required.*[7] It also has the remarkable property that an *individual* L-value (which need not be an integer) can be found, without computing a range of L-values. No other method in the literature has this property. Program KLEIN illustrates[10] this feature,

which is useful for relativistic calculations in which the effective L-values for each angular momentum channel are not integer-spaced. The algorithm consists in combining the ratio $F'_L/F_L \equiv f \equiv$ CF1 with the ratio $H'_L/H_L \equiv p + iq \equiv$ CF2 and the Wronskian (31), to solve, first for $F_L(\eta, x)$, and then for G_L, F'_L and G'_L all at the same time. Naturally the Wronskian relation cannot then be used also as an independent check of the solution. The details are explained in several references.[10,23,4,7] In practice, the lowest L is chosen and CF2 is evaluated, almost invariably, for $L = 0$. The value f is obtained from CF1 evaluated for the highest L required, followed by a downwards recurrence. The functions G_L and G'_L are found by upward recurrence from G_0 and G'_0.

In order to obtain CF1 we proceed as in §3. From (28), which is satisfied by $F_n(x)$, we see that the logarithmic derivative is given by,

$$\frac{F'_n}{F_n} = S_{n+1} - \frac{F_{n+1}}{F_n} \tag{32}$$

and from (25) a continued fraction for the ratio of successive orders can be derived[6,7] which is analogous to (22). Combining it with (32) we find,

$$\mathbf{CF1}: \quad f \equiv \frac{F'_n}{F_n} = S_{n+1} - \frac{R^2_{n+1}}{T_{n+1} -} \frac{R^2_{n+2}}{T_{n+2} -} \ldots \frac{R^2_k}{T_k - \ldots} \tag{33}$$

Eventually T_k will become large enough so that the value of f can be determined by terminating the evaluation at the k^{th} step, as in (24). To recover the specialised equation (22), set $\eta = 0$, *i.e.* $R^2_k = 1$ and $T_k = (2k+1)/x$, and take the derivatives of (5), which turns S_{n+1} into S_n.

A major feature of Steed's algorithm is the conversion of two (slow) asymptotically convergent series, for $H_L(\eta, x)$ and its derivative, into a ratio which could be expressed as a rapidly convergent continued fraction. This second continued fraction, CF2, is derived in ref.[6,7] and reads:

$$\mathbf{CF2}: \quad p + iq \equiv i(1 - \eta/x) + ix^{-1} \frac{ab}{2(x - \eta + i) +} \frac{(a+1)(b+1)}{2(x - \eta + 2i) + \ldots} \tag{34}$$

where $a = i\eta - L$ and $b = i\eta + L + 1$. It may also be computed by the L–T algorithm. For the case when x becomes smaller than x_L, eqn(2), then p loses accuracy relative to q. The reasons are explained in refs[7,10] and are discussed by Nesbet,[8] who suggests an alternative algorithm to minimise the difficulty. In practice it is only of minor concern when $\eta < 0$ because of (2), and because the minimum L needed is zero.

7. The Program COUL90

The new version of the COULFG program, COUL90, follows the logical flow of its predecessor[4] and differs, first, in the choice of the L–T algorithm for evaluating CF1, and secondly in adopting FORTRAN 77 conventions. Assume that the minimum L-value required is xm (not necessarily an integer) and the maximum is LRANGE + xm. (Except in rare circumstances[32] it is computationally desirable to set $xm = 0.0$. A serious loss of accuracy is possible when xm is chosen large enough that the required x-value is well below the turning point, x_{xm}.) CF1 is calculated for LRANGE + xm and then downward recurrence using (26), (27) carries the relative values of $F_n(\eta, x)$ and $F'_n(\eta, x)$ from $n =$ LRANGE $+ xm - 1$ to $n = xm$. The fraction CF2 is calculated at angular momentum xm and all the lowest-order functions are found, with the help of (31). Equations (29) and (30) are used to obtain all higher L-values, $xm + 1, xm + 2,$ ... , xm+LRANGE.

The CALLing argument list is:
CALL COUL90 (X, ETA, XLMIN,LRANGE, F,G,FP,GP, KFN, IFAIL)
where X, ETA, XLMIN are double-precision variables; LRANGE is the integer number of additional L-values required; F,G,FP,GP are double-precision arrays dimensioned at least (0:LRANGE); KFN selects the particular function (0 for Coulomb, 1 for spherical Bessel, and 2 for cylindrical Bessel); and IFAIL is an integer which is set to zero after a successful computation (and should be checked by the user). Generally XLMIN = xm, the minimum L-value, will lie between 0.0 and 1.0 (most usually 0.0) and LRANGE will just be the maximum L-value. To compute the oscillating Airy function, for example, then XLMIN $= -1/6$, LRANGE = 0 and a suitable[23] normalising constant is used. It is an easy matter to vary the calculation (as was done in the COULFG program) to obtain the various Bessel functions, and KFN parameter in the argument list allows for this. In the earlier code, provision was made for a mode choice, whereby storage could be saved by not storing the derivatives. This option has been removed, since most matching is formulated in terms of the logarithmic derivatives, and earlier core memory restrictions are no longer a factor.

The code produces results which are virtually the same as the test cases for COULFG,[4] to which reference should be made for the details. Minor differences can be traced to whether the compiler is set to optimise and if it is set to truncate, chop or randomise rounding. Selected test data are given in the next section.

8. Test calculations

The brief table contains values of the Coulomb functions calculated with the subroutine COUL90 for $x = 20$, 200 with $\eta = -0.50$, 0.00, 0.50 and for $x =$

1.0, 30.0, 1000.0, with $\eta = -5.2$ and each calculation for the range of orders, $L = 0 - 50$. Examples of a few spherical Bessel (j, y) and cylindrical Bessel functions (J, Y) are also given. They agree with the relevant codes SBESJY (j, y) and RICBES (xj, xy) and the values in Chapter 10 of Abramowitz and Stegun.[13] An extensive table incorporating most of the tests in ref.[4] and also the complete listings of the programs is to be found on the accompanying disc.

The last two calculations show the effect of taking xm too large. Six decimal places of accuracy are lost in $J_5(1.0)$ *etc* when $xm = 5.0$ compared with the values when $xm = 0.0$. The turning–point value is at $x = 5$ which is well outside the argument of $x = 1.0$.

Acknowledgements

This work was completed while the author was on sabbatical leave at the University of Auckland. He is most appreciative of the help and welcome that he received from the Physics Department there. The comments of Dr. I.J.Thompson on a draft version are also appreciated.

References

1. A. R. Curtis; *Coulomb Wave Functions* vol.11 *Royal Soc. Math. Tables* (Cambridge University Press, London, 1964) 9–25
2. M. J. Seaton; *Comp. Phys. Comm.* **25** (1982) 87–95; *ibid* **32** (1984) 115–119
3. K. L. Bell and N. S. Scott; *Comp. Phys. Comm.* **20** (1980) 447–458
4. A. R. Barnett; *Comp. Phys. Comm.* **27** (1982) 147–166 (program COULFG)
5. C. Bardin, Y. Dandeu, L. Gauthier, J. Guillermin, T. Lena, J.-M. Pernet, H. H. Wolter and T. Tamura; *Comp. Phys. Comp.* **3** (1972) 73–87
6. A. R. Barnett, D. H. Feng, J. W. Steed and L. J. B. Goldfarb; *Comp. Phys. Comp.* **8** (1974) 377–395 (program RCWFN)
7. A. R. Barnett; *J. Comp. Phys.* **46** (1982) 171–188
8. R. K. Nesbet; *Comp. Phys. Comm.* **32** (1984) 341–347
9. C. J. Noble and I. J. Thompson; *Comp. Phys. Comm.* **33** (1984) 413–419 (program COULN)
10. A. R. Barnett; *Comp. Phys. Comm.* **24** (1981) 141–159 (program KLEIN)
11. I. J. Thompson and A. R. Barnett; *Comp. Phys. Comm.* **36** (1985) 363–372 (program COULCC); *ibid* **47** (1987) 245–257 (program BESSCC)
12. I. J. Thompson and A. R. Barnett; *J. Comp. Phys.* **64** (1986) 490–509
13. H. A. Antosiewicz (Ch. 10) and M. Abramowitz (Ch.14) *Handbook of Mathematical Functions*, eds. M. Abramowitz and I. Stegun; (Nat. Bur. Stds., New York, 1964)
14. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling *Numerical Recipes: The Art of Scientific Computing* (Cambridge U. Press, New York, 1988)
15. S. E. Koonin; *Computational Physics* (Benjamin/Cummings 1985)
16. E. Gillman and H. R. Fiebig; *Comput. Phys.* **2 (1)** (1988) 62–72
17. W. J. Lentz; *Appl. Opt.* **15** (1975) 668–671
18. W. J. Wiscombe; *Appl. Opt.* **19** (1980) 1505–1509
19. W. D. Ross; *Appl. Opt.* **11** (1972) 1919–1923
20. C. G. van der Laan and N. M. Temme; *Calculation of special functions: the gamma function, the exponential integrals and error-like functions* (CWI tract 10 Mathematisch Centrum 1984)
21. P. Wynn; *Proc. K. Ned. Akad. Wet.* Ser. **A65** (1962) 127–148
22. J. Wimp; *Computation with Recurrence Relations* (Pitman, London, 1984)
23. A. R. Barnett; *Comp. Phys. Comm.* **21** (1981) 297–314
24. W. Gautschi; *SIAM Review* **9** (1967) 24–82
25. T. Jaaskelainen and J. Ruuskanen; *Appl. Opt.* **20** (1981) 3289–3290
26. W. J. Lentz; *Comput. Phys.* **4** (1990) 403–407
27. W. Gautschi and J. Slavik; *Math. Comp.* **32** (1978) 865–875
28. L. C. Welch; *Comput. Phys.* **2 (5)** (1988) 65–75
29. D. M. O'Brien; *J. Comp. Phys.* **36** (1980) 128–132
30. C. E. Fröberg; *Rev. Mod. Phys,* **27** (1955) 399–411
31. A. J. Strecock and J. A. Gregory; *Math. Comp.* **26** (1972) 955–975
32. D. H. Feng and A. R. Barnett; *Comp. Phys. Comm.* **11** (1976) 401–420

33. W. J. Cody and W. Waite; *Software Manual for the Elementary Functions* (Prentice Hall Series in Computational Mathematics 1980)
34. W. H. Press and S. A. Teukolsky; *Comput. Phys.* **5 (3)** (1991) in press