Algorithms and Codes for the Macdonald Function: Recent Progress and Comparisons

B.R. Fabijonas, [a] D.W. Lozier, [b] J.M. Rappoport [c]

[a] *Department of Mathematics, Southern Methodist University, Dallas, TX 75275-0156, USA*

[b] *Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg MD 20899-8910, USA*

[c] *Russian Academy of Sciences, Vlasov Street, Building 27, Apt. 8, Moscow, 117335, Russian Federation*

Abstract

The modified Bessel function $K_{i\nu}(x)$, also known as the *Macdonald function*, finds application in the Kontorovich-Lebedev integral transform when x and ν are real and positive. In this paper, a comparison of three codes for computing this function is made. These codes differ in algorithmic approach, timing, and regions of validity. One of them can be tested independently of the other two through Wronskian checks, and therefore is used as a standard against which the others are compared.

Key words: Macdonald functions, Bessel functions, computation of special functions

1 Introduction

Before the advent of desktop computers, the reference standard for special function values consisted of a large number of published tables. For a long time, such tables were crucial for computation. For example, over half of the well-known *Handbook of Mathematical Functions* [1], written in 1964, is composed of such tables, together with instructions for interpolation. As computers became commonplace, the value of codes which generate special function values became clear. A significant number of codes were written in a variety of languages, mainly Fortran 77, during the mid-1980s and early 1990s aimed at responding to this need. Today, tables as an aid to computation are relics of the pre-computer era. Many have been replaced with codes that are adequate for most applications.

This replacement is far from complete. The survey paper [10] contains a list of functions for which no appropriate software exists. Modern programming languages such as C, Fortran 90, and most recently JAVA, are reshaping the scientific computing environment. Their capabilities far surpass their Fortran 77 and Pascal predecessors and allow for new, more powerful, and more accurate codes to be written.

The modified Bessel functions $I_\mu(z)$ and $K_\mu(z)$ appear as independent solutions to the differential equation

$$z^2\frac{d^2w}{dz^2}+z\frac{dw}{dz}-(z^2+\mu^2)w=0. \tag{1.1}$$

Our concern is with the second function, known as the *Macdonald function*, in the special case $z=x, \mu=i\nu$, and x,ν real and positive. $K_{i\nu}(x)$ stands out in the family of Bessel functions as a real-valued solution to Eq. (1.1). It is of great interest since it forms the kernel of the Kontorovich-Lebedev transform, which is used to solve differential equations in wedge-shaped geometries.

When $x>0$ and $\nu>0$, the integral representations

$$I_{i\nu}(x)=\frac{1}{2\pi i}\int_{\infty-i\pi}^{\infty+i\pi}e^{x\cosh(t)-i\nu t}dt \tag{1.2}$$

and

$$K_{i\nu}(x)=\frac{1}{2}\int_{-\infty}^{\infty}e^{-x\cosh(t)+i\nu t}dt \tag{1.3}$$

are valid. Since $I_{i\nu}(x)$ is complex, Dunster [3] introduced the real-valued function

$$L_{i\nu}(x)=\tfrac{1}{2}\left(I_{i\nu}(x)+I_{-i\nu}(x)\right). \tag{1.4}$$

The functions $K_{i\nu}(x)$ and $L_{i\nu}(x)$ form a numerically satisfactory pair of solutions of Eq. (1.1) in the sense of Miller [13]; see Temme [25].

In this paper, we examine three different codes which compute the Macdonald function. These codes differ in algorithmic approach and other characteristics. Because of its robustness, flexibility, and internal consistency, the newest one can serve as a *reference code* against which the accuracy and regions of validity of the others can be compared.

The reference code will be described fully in a forthcoming paper [6]. It computes $K_{i\nu}(x)$, $L_{i\nu}(x)$, and their derivatives by integrating Eq. (1.1) numerically along stable paths or by summing uniform asymptotic or power series expansions. We refer to this code as *Code F* and briefly outline the algorithm in Section 2. The second code [21,23] was developed for, and entered into, the numerical program library at Moscow State University in the 1980's. It uses a combination of power series expansions, the Lanczos tau approximating method, and numerical quadratures. We refer to this code as *Code R* and discuss its algorithm in Section 3. The third code [5], *Code E*, was kindly provided to us by the author. It appeared as part of a code which computes Kontorovich-Lebedev transforms. It evaluates a modification of the integral representation Eq. (1.3) using quadrature routines available in the NAG [1] library [16]; see Ref. [5] for details. Section 4 contains a comparison of these codes against one another, and Section 5 concludes with a discussion and views towards the future.

2 Code F

Code F maximizes the use of uniform asymptotic expansions to compute the function $K_{i\nu}(x)$ and its derivative. In regions where the expansions do not yield sufficiently accurate results, it computes the function using Taylor series integration of Eq. (1.1) along stable integration paths. Since $x = 0$ is an accumulation point for the zeros of $K_{i\nu}(x)$, the oscillations for small values of x will be so dense that the integration will fail. In this case, Code F sums power series expansions which capture the oscillatory nature of the function. A detailed description of these regions and their determination can be found in Ref. [6].

2.1 Uniform asymptotic expansions

For x and/or ν large, Code F evaluates uniform asymptotic expansions for the function and its derivative. Using the notation of Temme [26], the expansion is

$$K_{i\nu}(\nu x) = \frac{\pi e^{-\nu\pi/2}}{\nu^{1/3}}\phi(\zeta)\Big\{ \mathrm{Ai}(-\nu^{2/3}\zeta)A_\nu(-\zeta) + \nu^{-4/3}\,\mathrm{Ai}'(-\nu^{2/3}\zeta)B_\nu(-\zeta)\Big\}, \quad (2.1)$$

[1] The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology.

as $\nu \to \infty$, uniformly with respect to $x \in (0, \infty)$, where ζ is given by

$$\begin{aligned} \frac{2}{3}\Big[\zeta(x)\Big]^{3/2} &= \ln\left\{\frac{1+(1-x^2)^{1/2}}{x}\right\} - (1-x^2)^{1/2}, \qquad & 0 < x \le 1, \\ \frac{2}{3}\Big[-\zeta(x)\Big]^{3/2} &= \sqrt{x^2-1} - \arccos\frac{1}{x}, & x \ge 1, \end{aligned} \tag{2.2}$$

and

$$\phi(\zeta) = \left(\frac{4\zeta}{1-x^2}\right)^{1/4}.$$

The function $\mathrm{Ai}(x)$ is the standard Airy function, and the 'slowly varying' parts $A_\nu(-\zeta)$ and $B_\nu(-\zeta)$ represent the expansions

$$A_\nu(-\zeta) \sim \sum_{s=0}^{\infty}(-1)^s \frac{a_s(\zeta)}{\nu^{2s}} \quad \text{and} \quad B_\nu(-\zeta) \sim \sum_{s=0}^{\infty}(-1)^s \frac{b_s(\zeta)}{\nu^{2s}}. \tag{2.3}$$

The coefficients $a_s(\zeta)$ and $b_s(\zeta)$ have specific representations in terms of integral-recurrence relations. We refer the reader to Ref. [6] for details. The expansion in Eq. (2.1) was first found by Balogh [2] and later refined by Dunster [3].

The algorithm in Section 2.4 below will require the computation of $K'_{i\nu}(x)$ using a uniform asymptotic expansion. Therefore, the derivation of such an expansion is outlined here. As was done in Ref. [17], one can differentiate the expression in Eq. (2.1) with respect to ζ to obtain

$$\begin{aligned} K'_{i\nu}(\nu x) = -\frac{\pi e^{-\nu\pi/2}}{\nu^{4/3}}\hat{\phi}(\zeta)\Bigg\{ & \mathrm{Ai}(-\nu^{2/3}\zeta)C_\nu(-\zeta) \\ & + \nu^{2/3}\,\mathrm{Ai}'(-\nu^{2/3}\zeta)D_\nu(-\zeta)\Bigg\} \end{aligned} \tag{2.4}$$

as $\nu \to \infty$, uniformly with respect to $x \in (0, \infty)$. Here

$$\hat{\phi}(\zeta) = -\frac{d\zeta}{dx}\phi(\zeta) = \frac{2}{x\phi(\zeta)}$$

and the notation $C_\nu(-\zeta)$ and $D_\nu(-\zeta)$ has been introduced to represent the terms

$$\begin{aligned} C_\nu(-\zeta) &= \chi(\zeta)A_\nu(-\zeta) + \frac{d}{d\zeta}(A_\nu(-\zeta)) + \zeta B_\nu(-\zeta), \\ D_\nu(-\zeta) &= \nu^{-2}\chi(\zeta)B_\nu(-\zeta) - A_\nu(-\zeta) + \nu^{-2}\frac{d}{d\zeta}(B_\nu(-\zeta)), \end{aligned}$$

where

$$\chi(\zeta) = \frac{\phi'(\zeta)}{\phi(\zeta)} = \frac{4 - x^2\phi^6(\zeta)}{16\zeta}.$$

Using the expansions in Eq. (2.3) in the above equations, one finds

$$C_\nu(-\zeta) \sim \sum_{s=0}^{\infty}(-1)^s \frac{c_s(\zeta)}{\nu^{2s}}, \quad D_\nu(-\zeta) \sim \sum_{s=0}^{\infty}(-1)^{s+1}\frac{d_s(\zeta)}{\nu^{2s}}, \tag{2.5}$$

where, upon equating like powers of ν,

$$\begin{aligned} c_s(\zeta) &= \chi(\zeta)a_s(\zeta) + a_s'(\zeta) + \zeta b_s(\zeta) \qquad s = 0, 1, 2, \ldots, \\ d_s(\zeta) &= \chi(\zeta)b_{s-1}(\zeta) + a_s(\zeta) + b_{s-1}'(\zeta) \qquad s = 1, 2, \ldots, \end{aligned}$$

with $d_0 = a_0$. Again, the reader is referred to Ref. [6] for details.

The uniform asymptotic expansions also require the computation of the Airy functions $\mathrm{Ai}(x)$ and its derivative. There are several codes which compute these functions. Code F uses a new code [7] written in Fortran 90.

2.2 Uniform asymptotic expansions at the turning point

The uniform asymptotic expansions given above are very powerful in that they are valid for a large range of variables. However, they become numerically unstable near the turning point $x = 1$, or equivalently, $\zeta = 0$. Two algorithms to compensate for this instability are given in Temme [26], where the author considered the Airy-type uniform asymptotic expansions for the unmodified Bessel functions $J_\nu(x)$, $Y_\nu(x)$, and their derivatives.

The first algorithm of Ref. [26] expands the terms $a_s(\zeta)$, $b_s(\zeta)$, $c_s(\zeta)$, $d_s(\zeta)$, $\phi(\zeta)$, $\hat{\phi}(\zeta)$, and $\chi(\zeta)$ in Maclaurin series of the variable $\eta = 2^{-1/3}\zeta$. This method requires storing a large number of coefficients which can be generated exactly in rational form using a computer algebra system and stored in floating point form. The second algorithm reorders the expansions for the slowly varying parts in powers of ζ. The coefficients are computed in floating point using an iteration process which converges quickly. The obvious advantage of this algorithm is that the number of terms that must be manually placed in storage with the routine is greatly reduced. Code F uses the first algorithm because of its simplicity. In principle, the second algorithm could be adapted for use in Eqs. (2.3) and (2.5), though one needs to be mindful that these expansions differ in form from those in Ref. [26].

2.3 Power series

A power series expansion for the Macdonald function is

$$K_{i\nu}(x) = -\left(\frac{\nu\pi}{\sinh(\nu\pi)}\right)^{1/2} \sum_{s=0}^{\infty} h_s(\nu)\left(\frac{x^2}{4}\right)^s \sin(\nu \ln(x/2) - \phi_{\nu,s}), \qquad (2.6)$$

where

$$h_s(\nu) = \{s![(\nu^2)(1^2+\nu^2)\cdots(s^2+\nu^2)]^{1/2}\}^{-1} \qquad (2.7)$$

and

$$\phi_{\nu,s} = \arg\left\{\Gamma(1+s+i\nu)\right\}. \qquad (2.8)$$

The functions $h_s(\nu)$ and $\phi_{\nu,s}$ are continuous in ν and $\phi_{\nu,s} \to 0$ as $\nu \to 0$; cf. Dunster [3]. There is a typographical error in Ref. [3] in the expansion of $K_{i\nu}(x)$ where it should read $(x^2/4)^s$ rather than $(x^2/4)^2$. The computation of $\phi_{\nu,s}$ is discussed in Ref. [6].

2.4 Taylor series integration of the ODE

In the final region, Code F computes $K_{i\nu}(x)$ and its derivative using Taylor series integration of Eq. (1.1) in the framework of an initial value problem along rays parallel to the x axis as described by Lozier and Olver [11]. This integration must be carried out in a stable manner, which means that the wanted solution must grow at least as fast as all independent solutions. The error term will be a linear combination of the wanted solution and a linearly independent solution. Since at infinity $|K_{i\nu}(x)|$ is exponentially small and all linearly independent solutions are exponentially large in modulus, the desired direction for integration is towards the origin for $K_{i\nu}(x)$. Thus, the contribution of the unwanted solution to the error term decays exponentially in the chosen direction of integration. The initial values at sufficiently large values of x are computed from the uniform asymptotic expansions discussed in Section 2.1.

The integration path is divided into $P+1$ points labeled $x_0, x_1, \ldots, x_P$, where x_0 is the initial point of the integration path and x_P is the terminal point at which the function value is desired. Pairs of independent solutions $p_j(x)$ and $q_j(x)$ are constructed for each interval $x_{j-1}x_j$ with the initial conditions

$$p_j(x_{j-1}) = 1 \quad p_j'(x_{j-1}) = 0 \quad q_j(x_{j-1}) = 0 \quad q_j'(x_{j-1}) = 1. \tag{2.9}$$

Then,

$$\begin{pmatrix} y(x_P) \\ y'(x_P) \end{pmatrix} = A_P A_{P-1} A_{P-2} \cdots A_1 \begin{pmatrix} y(x_0) \\ y'(x_0) \end{pmatrix}, \tag{2.10}$$

where A_j denotes the matrix

$$A_j = \begin{pmatrix} p_j(x_j) & q_j(x_j) \\ p_j'(x_j) & q_j'(x_j) \end{pmatrix}, \qquad j = 1, 2, \ldots, P \tag{2.11}$$

and $y(x)$ is the solution to Eq. (1.1) for a fixed value of ν. The elements of A_j are computed from the Taylor series expansion of $p_j(x)$ and $q_j(x)$ at $x = x_{j-1}$.

3 Code R

This code computes $K_{i\nu}(x)$ by summing a power series expansion or by finding an approximate solution to Eq. (1.1) by using the Lanczos tau-method or by using quadratures. Code R originally was used to compute $K_{i\nu}(x)$ only on the square $0 < x < 10$, $0 < \nu < 10$. A figure is shown in Ref. [27] which indicates the regions in which each algorithm was used.

3.1 Power series

The power series (2.6) can be reexpressed in the form

$$K_{i\nu}(x) = \left(\frac{\pi}{\nu \sinh(\nu\pi)}\right)^{1/2} \Big(A(\nu, x)\sin(\theta(\nu, x)) + B(\nu, x)\cos(\theta(\nu, x))\Big) \tag{3.1}$$

for $x > 0$, where

$$\theta(\nu, x) = \nu \ln \frac{x}{2} - \arg \Gamma(i\nu), \tag{3.2}$$

$$A(\nu, x) = \sum_{k=0}^{\infty} a_k(\nu, x), \qquad B(\nu, x) = \sum_{k=0}^{\infty} b_k(\nu, x). \tag{3.3}$$

The terms $a_k(\nu, x)$, $b_k(\nu, x)$ are generated by the recurrences

$$a_{-1} = 0, \qquad b_0 = 1, \tag{3.4}$$
$$a_k = a_{k-1}m_k + b_k n_k, \qquad b_{k+1} = b_k m_k - a_{k-1} n_k, \quad k = 0, 1, 2, \ldots, \tag{3.5}$$

where, for $k = 0, 1, 2, \ldots$,

$$m_k = \frac{x^2}{4((k+1)^2 + \nu^2)}, \quad n_k = \frac{\nu x^2}{4(k+1)((k+1)^2 + \nu^2)}. \tag{3.6}$$

For details, see Refs. [15,27]. This expansion converges for $0 < x < \infty$ and $0 < \nu < \infty$ but is computationally efficient for only small and moderate values of x/ν.

3.2 The Lanczos tau-method

Consider Eq. (1.1) on the interval $x \in [1, \infty)$. Changing variables by

$$y(x) = \left(\frac{\pi}{2x}\right)^{1/2} e^{-x} V(x),$$

and moving the singular point at infinity to the origin by the transformation $\gamma = 1/x$, Eq. (1.1) becomes

$$\gamma^2 v''(\gamma) + 2(\gamma + 1)v'(\gamma) + (\tfrac{1}{4} + \nu^2)v(\gamma) = 0, \tag{3.7}$$

where $v(\gamma) = V(x)$. An initial condition for $v(\gamma)$ corresponds to a condition at infinity for $K_{i\nu}(x)$, which is given by

$$v(0) = \lim_{x\to\infty} e^x \sqrt{\frac{2x}{\pi}} K_{i\nu}(x) = 1.$$

This follows from Poincaré expansions for $K_{i\nu}(x)$. See, for example, Ref. [12]. Taking two integrals of Eq. (3.7), integrating by parts, and using the identity

$$\int_0^\gamma \int_0^\xi v(t)\, dt\, d\xi = \int_0^\gamma (\gamma - \xi)v(\xi)\, d\xi, \tag{3.8}$$

a Volterra-type integral equation for $v(\gamma)$ is obtained, namely

$$\gamma^2 v(\gamma) = \int_0^\gamma \Big[(\tfrac{9}{4} + \nu^2)x - (\tfrac{1}{4} + \nu^2)\gamma - 2\Big] v(x)\, dx + 2\gamma. \tag{3.9}$$

The problem now is formulated so that one can use the tau-method [19] and the approximation method [4] to construct a sequence of approximate solutions to

Eq. (3.9) denoted by $v_n(\gamma)$. The elements of the sequence $v_n(\gamma)$ are polynomials of degree n and are solutions to the following equation:

$$\gamma^2 v_n(\gamma) = \int_0^\gamma \left[(\tfrac{9}{4}+\nu^2)x - (\tfrac{1}{4}+\nu^2)\gamma - 2\right] v_n(x)\,dx + 2\gamma \\ + \tau_1 T^*_{n+1}(\gamma) + \tau_2 T^*_{n+2}(\gamma), \quad (3.10)$$

where $T^*_j(\gamma)$ is the jth shifted Chebyshev polynomial and τ_1, τ_2 are coefficients to be determined.

The tau-method constructs the solutions $v_n(\gamma)$ by first constructing a set of *canonical polynomials* for Eq. (3.9), denoted by $Q_m(x)$, where m indicates the degree of the polynomial. These polynomials are solutions of the equation

$$\gamma^2 Q_m(\gamma) = \int_0^\gamma [(\tfrac{9}{4}+\nu^2)x - (\tfrac{1}{4}+\nu^2)\gamma - 2] Q_m(x)\,dx + 2\gamma Q_m(0) + \gamma^{m+2}. \quad (3.11)$$

See Refs. [19,20]. These canonical polynomials assist in the transition from nth to $(n+1)$st approximation. By writing the canonical polynomials in the form

$$Q_m(\gamma) = \sum_{i=0}^{m} b_{mi}\gamma^i, \quad (3.12)$$

a (stable) first-order linear recurrence for the coefficients b_{mi} is determined:

$$b_{mm} = \frac{1}{f_m}, \qquad b_{mi} = \frac{-2}{(i+2)f_i} b_{m,i+1}, \qquad i = m-1, m-2, \ldots, 0,$$

where

$$f_i = 1 - \frac{9/4+\nu^2}{i+2} + \frac{1/4+\nu^2}{i+1}, \qquad i = 0, 1, 2, \ldots, m.$$

Then, the tau-method constructs the polynomial solution to Eq. (3.10) by expanding $v_n(\gamma)$ in terms of the above constructed canonical polynomials:

$$v_n(\gamma) = \tau_1 \sum_{m=0}^{n-1} C(m+2, n+1) Q_m(\gamma) + \tau_2 \sum_{m=0}^{n} C(m+2, n+2) Q_m(\gamma), \quad (3.13)$$

where

$$C(m,j) = (-1)^{j+m} 2^{2m-1} \left[2\tbinom{j+m}{j-m} - \tbinom{j+m-1}{j-m}\right],$$

are the coefficients of the shifted Chebyshev polynomials, that is,

$$T_j^*(\gamma) = \sum_{m=0}^{j} C(m,j)\gamma^m.$$

Finally, τ_1 and τ_2 are obtained by inserting the solution form (3.13) into Eq. (3.10):

$$\begin{aligned} \tau_1 C(0,n+1) + \tau_2 C(0,n+2) &= 0, \\ -2v_n(0) + 2 + \tau_1 C(1,n+1) + \tau_2 C(1,n+2) &= 0. \end{aligned} \tag{3.14}$$

Summarizing,

$$K_{i\nu}(x) = e^{-x}\sqrt{\frac{\pi}{2x}}\left[\sum_{m=0}^{n-1}\left\{\tau_1 C(m+2,n+1) + \tau_2 C(m+2,n+2)\right\}Q_m\left(\frac{1}{x}\right)\right. \\ \left. + \tau_2 C(n+2,n+2)Q_n\left(\frac{1}{x}\right) + R_n\right], \tag{3.15}$$

valid on the interval $1 \le x < \infty$, where R_n is the remainder. In the code, n was chosen by numerical experimentation to satisfy $|R_n| < 10^{-16}$ within the region where the method was applied.

3.3 Numerical quadrature

Different quadrature formulas (Filon's, Gauss-Legendre, quadrature of the trapezoidal kind) were considered. It was determined that the best accuracy and speed were achieved with quadrature formulas of trapezoidal kind.

An integral representation of the modified Bessel function $K_{i\nu}(x)$ is

$$K_{i\nu}(x) = \int_0^\infty e^{-x\cosh t}\cos(\nu t)dt; \tag{3.16}$$

cf. Eq. (1.3). Because of the rapidly decreasing integrand for increasing t, it is possible to truncate these integrals while maintaining the necessary precision. Thus, Code R uses the truncated integral

$$I = \int_0^b e^{-x\cosh t}\cos(\nu t)dt, \tag{3.17}$$

where the upper limit of integration b is determined from the condition

$$e^{x(1-\cosh b)} = 10^{-N};$$

the code sets $N = 16$.

The trapezoidal rule gives

$$I(h) = h\left(0.5e^{-x} + \sum_{j=1}^{k} e^{-x\cosh(jh)}\cos(\nu jh)\right), \tag{3.18}$$

where $k = [b/h]$. This could be implemented by starting with an initial $h = h_0$, generating $I(h)$ for $h = h_0, h_0/2, h_0/4, \ldots$ until the relative difference between successive approximations is smaller than 10^{-N-1}, but this procedure may cost many iterations, depending on the value of x. This difficulty was avoided in Ref. [9] by estimating the relative error of quadrature formula (3.18) as

$$|\ \Delta I(h)\ | \cong 10^{-M(\pi^2/h - \pi\nu/2)}/K_a, \tag{3.19}$$

where $M = \log_{10} e$ and K_a is the value of $K_{i\nu}(x)$ computed from the first term in its Poincaré asymptotic expansion. Requiring $|\ \Delta I(h)\ | \leq 10^{-N}$, h is chosen to satisfy

$$h \leq \pi^2/(N/M + \pi\nu/2 - \ln(K_a)). \tag{3.20}$$

The code sets $N = 16$. The method is valid for all x and ν but becomes computationally inefficient as $x/\nu \to 0$; c.f. Ref. [9].

4 Comparisons

The first purpose of this section is to establish the validity of the reference Code F. The second is to compare the other two codes to the reference code. In all comparisons the error measure will be the *relative precision* of two real numbers x and y, defined as

$$\mathrm{rp}(x, y) = |\ln(x/y)|. \tag{4.1}$$

This is a symmetric variation of conventional relative error, applicable when x and y are nonzero and have the same sign; cf. Lozier [10] and Olver [18]. As a rule in floating-point computation, the rp will increase as zeros are approached. In such cases, only absolute precision can be maintained.

A reliable test for a numerically satisfactory pair of solutions of a linear second-order ordinary differential equation is verification of Wronskians. Of the codes considered here, this method is available only for Code F. The Wronskian to be used is

$$\mathcal{W}\{K_{i\nu}(x), L_{i\nu}(x)\} = 1/x. \tag{4.2}$$

Because $K_{i\nu}(x)$ and $L_{i\nu}(x)$ form a numerically satisfactory pair of solutions, the rp of Eq. (4.2) should be constant for all x and ν. This rp is a good measure

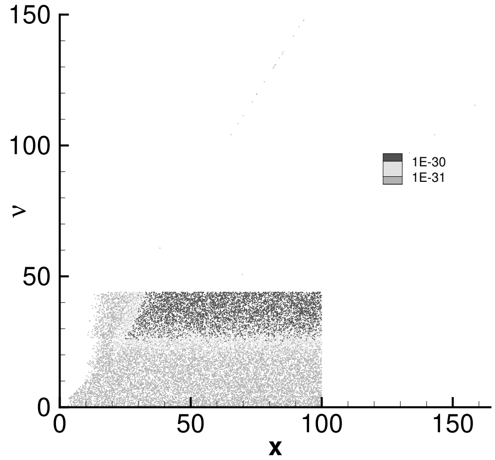


Fig. 1. The relative precision (rp) of the Wronskian of $K_{i\nu}(x)$ and $L_{i\nu}(x)$ computed by Code F in 128-bit precision. The white parts of the region $0 < x, \nu \leq 150$ correspond to points where the rp is less than 10^{-32}. The maximum rp is 3.0^{-26}.

of the rp of the two functions and their derivatives, except near a zero of one of them.

Code F was designed to be able to compute the functions $K_{i\nu}(x)$ and $L_{i\nu}(x)$ in various precisions. Using a Fortran 90 extension offered by many compilers, it was run with a wordlength of 128 bits. The significand length was 113, corresponding to an rp of 2^{-112}, or $10^{-33.7}$, approximately. Figure 1 shows points for which the rp of Eq. (4.2) exceeds 10^{-32} on a random grid. The figure is a good measure also of the rp of $K_{i\nu}(x)$ for points ν, x that are not "close" to a zero of $K_{i\nu}(x)$. The largest errors occur near the line $\nu = 44$ at the top of the integration region. This is attributed to error amplification due to argument reduction in computing trigonometric functions used in the power series expansion for $L_{i\nu}(x)$. This expansion provides the starting values for integration and the errors propagate into the integration region.

As an aside, the apparent boundaries in the figure correspond, in part, to the computational subdomains of Code F. For parameter values which fall in the quadrilateral $\nu < x \leq 100$, $0 < \nu \leq 44$, the functions are computed using integration of the ODE. For parameter values in the triangle $0 < x \leq \nu \leq 44$, Code F uses power series expansions. Outside these regions, Code F sums the uniform asymptotic expansions, and in particular uses the method described in Section 2.2 for $0.72\nu \leq x \leq 1.59\nu$.

The comparison of the other codes with Code F is carried out in the following way. First, a random grid is generated in the precision of the code that is to be tested, in this case double precision on a 32-bit machine (64 bits with 53-bit significand), corresponding to an rp of 2^{-52}, or $10^{-15.6}$, approximately. Next, function values are generated on this grid using Code F at twice the precision.

This amount of guarding precision is sufficient to provide a virtual guarantee that comparison values to full double-precision rp are available on the random grid, even in neighborhoods of finite zeros.

The upper and lower parts of Figure 2 show points for which the relative precision of Code R exceeds 10^{-14}. The upper part covers the region for which Code R was designed, $0 < \nu, x \leq 10$. The maximum rp is 4.6×10^{-9} which lies at $x = 0.13$, $\nu = 8.1$ near a zero of $K_{i\nu}(x)$. The much larger region covered in the lower part of the figure, $0 < \nu, x \leq 150$, is included for comparison with the other codes; see Figs. 1 and 3. It can be noted that Code R performs well in a substantially larger region than called for by its design, and that further improvements of Code R are possible with a sufficient expenditure of effort. The stairlike boundary in the upper part of Fig. 2 corresponds, in part, to the different computational domains: to the left, Code R uses power series summation; to the right and below $\nu = 4.2$, the Lanczos tau-method; and in the final region, the numerical quadrature. The curves seen in the upper part of the figure for $x < \nu$ correspond to zeros of $K_{i\nu}(x)$. These curves indicate the difficulty Code R has in computing the function value near a zero, a feature that is common to many algorithms when executed in floating point arithmetic.

Figure 3 is similar to Figure 2, but for Code E. Again, we see the expected difficulty in computing the zeros of $K_{i\nu}(x)$. The accuracy of Code E is limited by the tolerances chosen in the NAG subroutine calls. In order to cover as large a part of the domain as possible, we use the accuracy of 10^{-14} on $0.75 < x \leq 150$, $0 < \nu \leq 150$. Computations in the strip $0 < x < 0.75$ require significantly reduced tolerances and is therefore omitted. We see that Code E returns values accurate to single precision in much of the square $0 < x, \nu \leq 10$, and that this accuracy is continued outside this square. In fact, the accuracy is doubled for large values of x and small ν. The maximum rp is 1.9×10^{-3} at $x = 1.2$, $\nu = 6.4$ near a zero of $K_{i\nu}(x)$.

5 Discussion and views towards the future

The purpose of a *reference code* is to provide a standard against which to compare other codes that compute the same functions. Speed of execution is secondary to accuracy, robustness and reliability. Ability to compute to high precision, in the current computing environment to at least quadruple precision, is essential. Overflow and underflow should be mitigated or eliminated. These attributes should apply to a very wide range of argument and parameter values, so that tests can be constructed to cover all eventualities. Coincidentally, a reference code can serve as a working code in applications where no superior alternative exists.

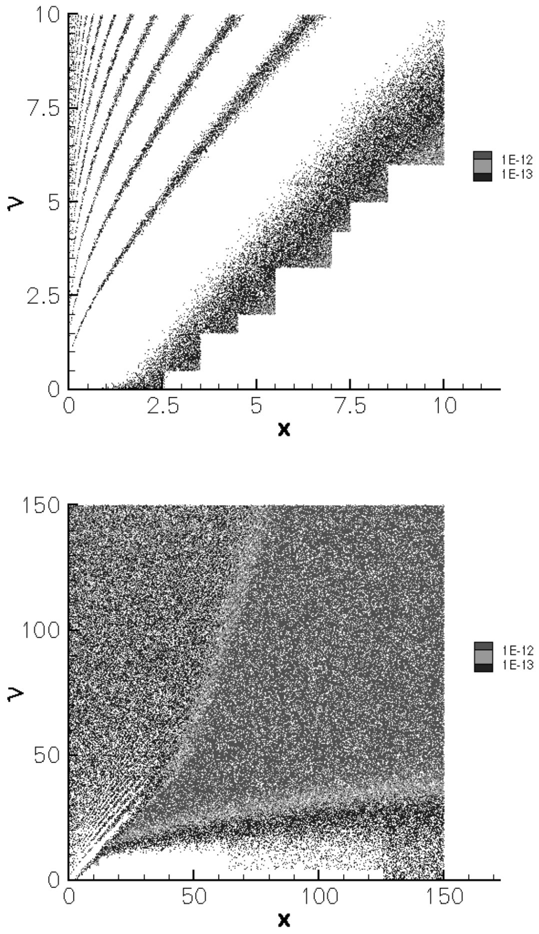
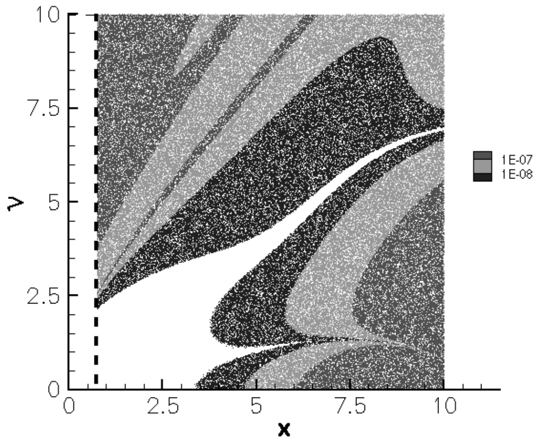


Fig. 2. The relative precision (rp) of Code R. The upper figure covers the region for which Code R was designed, $0 < x, \nu \leq 10$. The much larger region covered in the lower figure, $0 < x, \nu \leq 150$, is included only for comparison with Codes F and E (see Figs. 1 and 3). The white parts of the figures correspond to points where the rp is less than 10^{-14}.

We have found that graphical display of errors is a useful way of presenting code comparisons. It allows us to see at a glance whether or not a given code is suitable for a particular application. It also provides assistance in code development. For example, with detailed information about the location of the largest errors, we can examine the code to see if we can remove bugs or incorporate better algorithms.

Code F was designed to be a reference code, capable of generating function values in single, double and quadruple precision. The previous section demonstrates its value in assessing the capabilities of two codes that compute the Macdonald function $K_{i\nu}(x)$. From Fig. 2 we were able to see immediately that Code R, for example, meets its design criteria within the small region $0 < x, \nu \leq 10$, but also that it remains valid for much larger values of x if ν is



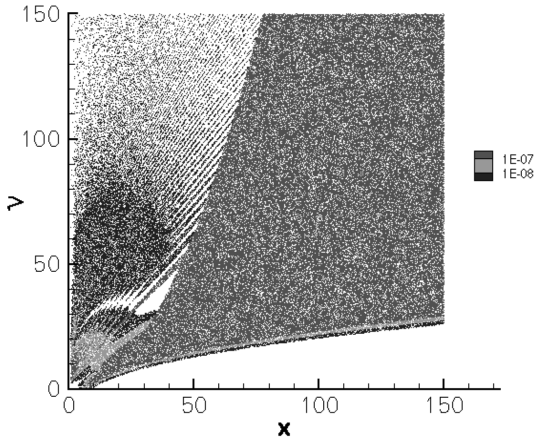


Fig. 3. The relative precision (rp) of Code E. The upper figure covers the region $0 < \nu \leq 10$ and $0.75 \leq x \leq 10$, for comparison with Code R. The lower figure covers the region $0 < \nu \leq 150$, $0.75 \leq x \leq 150$, for comparison with the other two codes. The white parts of the figures correspond to points where the rp is less than 10^{-9}.

relatively small, and vice versa. We were also able to observe the boundaries between subregions of $0 < x, \nu \leq 10$ in which different algorithms were used. Boundaries are often places where algorithms break down, and the larger errors to the left of the stairlike structure in the figure provides a guide to possible improvements in the code.

Figures 2 and 3 also show that Codes R and E cannot deal with computations of $K_{i\nu}(x)$ when x, ν are both large. Accordingly, for some applications it may be necessary to use the reference Code F. A comparison of the double precision speed of Codes F and R is shown in Figure 4. These timings are somewhat misleading in that Code R computes only the function $K_{i\nu}(x)$ whereas Code F computes both the function and its derivative simultaneously. Code R almost always is faster when $0 < x, \nu \leq 10$, and this advantage could be important in Kontorovich-Lebedev transforms. The timings were obtained

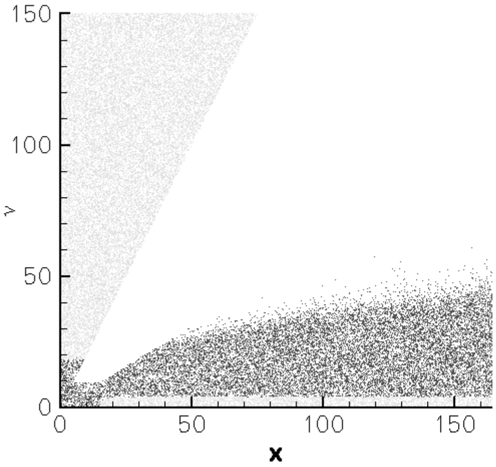


Fig. 4. Timing comparison. Code F was faster at light gray points, and Code R for the dark gray points. The white regions correspond to points where the rp of Code R exceeds 10^{-12}. In particular, Code F is faster in the thin region just above the x-axis for $x \geq 15$.

using the `STOPWATCH` timing program [14]. It will be a challenge for software developers to construct codes for $K_{i\nu}(x), L_{i\nu}(x)$ that are more efficient than the reference Code F.

Finally, it should be mentioned that work by Temme [24,25] has prepared the theoretical framework for yet another algorithm for computing $K_{i\nu}(x)$, $L_{i\nu}(x)$, and their derivatives by using the method of steepest descents and quadrature with the integral representations (1.2), (1.3). Work is in progress by Gil, Segura and Temme [8].

6 Acknowledgments

The authors thank F.W.J. Olver for fruitful discussions and U.T. Ehrenmark for providing his code for evaluation. The research described in this publication was made possible in large part by Award No. RM1-361 of the U.S. Civilian Research & Development Foundation for the Independent States of the Former Soviet Union (CRDF) [2] for two of the authors (DL and JR), and the Young Investigator Addendum to this award for the third (BF). Part of this work was conducted while the first author (BF) was a National Research Council Postdoctoral Fellow in the Information Technology Laboratory of the National Institute of Standards and Technology (NIST) and was funded accordingly. Two of the authors (BF and DL) are indebted to the Russian Academy of

[2] Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the CRDF.

Sciences and the Institute for Computer Aided Design in Moscow, Russia and one of them (JR) to the Mathematical and Computational Sciences Division of NIST in Washington, DC for their hospitality.

References

[1] M. Abramowitz, I. A. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, 10th Edition, US Government Printing Office, Washington DC, 1972.

[2] C. B. Balogh, Asymptotic expansions of the modified Bessel function of the third kind of imaginary order, *SIAM J. Appl. Math.* **15** (1967) 1315–1323.

[3] T. M. Dunster, Bessel functions of purely imaginary order, with an application to second-order linear differential equations having a large parameter, *SIAM J. Math. Anal.* **21** (1990) 995–1018.

[4] V. K. Dzyadyk, *Approximation methods for solutions of differential and integral equations*, VSP, Utrecht, 1995, translation of *Approksimatsionnye metody resheniya differentsialnykh i integralnykh uravneniĭ*, Naukova Dumka, Kiev, 1988.

[5] U. T. Ehrenmark, The numerical inversion of two classes of Kontorovich-Lebedev transforms by direct quadrature, *J. Comp. Appl. Math.* **61** (1995) 43–72.

[6] B. R. Fabijonas, Computing the real-valued modified Bessel functions of imaginary order by ODE integration, submitted.

[7] B. R. Fabijonas, D. W. Lozier, F. W. J. Olver, Algorithm xxx: Airy functions, Submitted.

[8] A. Gil, J. Segura, N. M. Temme, Evaluation of the modified Bessel function of the third kind of imaginary orders, *J. Comput. Phys.* **175** (2002) 398–411.

[9] T. Kiyono, S. Murashima, A method of evaluation of the function $K_{is}(x)$, *Mem. Fac. Engr. Kyoto Univ.* **35** (1973) 102–127.

[10] D. W. Lozier, Software needs in special functions, *J. Comput. Appl. Math.* **66** (1996) 345–358.

[11] D. W. Lozier, F. W. J. Olver, Airy and Bessel functions by parallel integration of ODEs, in: R. Sincovec, D. Keyes, M. Leuze, L. Petzold, D. Reed (Eds.), *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Vol. 2*, Society for Industrial and Applied Mathematics, 1993, pp. 530–538.

[12] W. Magnus, F. Oberhettinger, R. Soni, *Formulas and Theorems for the Special Functions of Mathematical Physics*, Springer-Verlag, Berlin, 1966.

[13] J. C. P. Miller, On the choice of standard solutions for a homogeneous linear differential equation of the second order, *Quart. J. Mech. Appl. Math.* **3** (1950) 225–235.

[14] W. Mitchell, StopWatch user's guide version 1.0, Tech. Rep. NISTIR 5971, Information Technology Laboratory, NIST (March 1997).

[15] A. F. Nikiforov, V. B. Uvarov, *The Special Functions of Mathematical Physics*, Birkhauser, 1988, translation of *Spetsialnye Funktsii Matematicheskoi Fiziki*, Nauka, Moscow, 1978.

[16] Numerical Algorithms Group, NAG FORTRAN Library Manual, Mark 14, Volume 12 (1993).

[17] F. W. J. Olver, Some new asymptotic expansions for Bessel functions of large orders, *Proc. Cambridge Philos. Soc.* **48** (1952) 414–427.

[18] F. W. J. Olver, A new approach to error arithmetic, *SIAM J. Numer. Anal.* **15** (1978) 368–393.

[19] E. L. Ortiz, Canonical polynomials in the Lanczos tau-method, in: B. Scaife (Ed.), *Studies in Numerical analysis*, Academic Press, 1974, pp. 73–93.

[20] J. M. Rappoport, The Tau method and the computation of special functions of complex parameter, preprint of *Vych. Tsentr. AN SSSR* (1978) 1-28 (in Russian).

[21] J. M. Rappoport, Application of the approximating method to the computation of modified Bessel functions of imaginary index, preprint of *Vych. Tsentr. AN SSSR* (1982) 1–22 (in Russian).

[22] J. M. Rappoport, A computational scheme for an approximating method, *Differents. Uravn.* **22** (1986) 2162–2166 (in Russian).

[23] J. M. Rappoport, The programs and some methods of the Macdonald function computation, preprint of *Otd. Vych. Matem. AN SSSR* (1990) 1–16 (in Russian).

[24] N. M. Temme, On the numerical evaluation of the modified Bessel function of the third kind, *J. Comp. Phys.* **17** (1975) 324–337.

[25] N. M. Temme, Steepest descent paths for integrals defining the modified Bessel functions of imaginary order, *Meth. Appl. Anal.* **1** (1994) 14–24.

[26] N. M. Temme, Numerical algorithms for uniform Airy-type asymptotic expansions, *Num. Alg.* **15** (1997) 207–225.

[27] M. I. Zhurina, L. N. Karmazina, The tables of the modified Bessel functions with imaginary index, *Vych. Tsentr. AN SSSR*, (1967) 1-342 (in Russian).