Interdisciplinary Information Sciences, Vol. 6, No. 1, pp. 67–74 (2000)

Numerical Calculation of the Elliptic Integrals of the First and Second Kinds with Complex Modulus

Tohru MORITA

Department of Computer Science, College of Engineering, Nihon University, Koriyama 963-8642, Japan

Received September 13, 1999; final version accepted December 13, 1999

Examination of convergence is given of a numerical procedure for calculating the elliptic integrals of the first and second kinds with complex modulus and their analytic continuations.

KEYWORDS: elliptic integral, numerical calculation, complex modulus, analytic continuation

1. Introduction

Bulirsch (1965) gave algorithms of calculating the elliptic integrals, which are based on the arithmetic-geometric mean or the Landen transform. Morita and Horiguchi (1973) studied the convergence of the algorithm for the complete elliptic integrals of the first and second kinds $K(k)$ and $E(k)$ for the case when the modulus k takes complex values. Morita (1978) then studied the convergence of the algorithm for their analytic continuation, in the Riemann sheet where the argument of the complementary modulus k' is between $-\pi$ and π, where $k' = \sqrt{1 - k^2}$.

Carlson (1979) presented an algorithm of calculating elliptic integrals, where he stated that a program based on the algorithm is applicable to complex values. In the book of "Numerical Recipes in C" by Press et al. (1992), the program given for the elliptic integrals is based on Carlson's algorithm. By that program, we can calculate the *incomplete* as well as the complete elliptic integrals by the same procedure. In order to check its utility for complex modulus, the present author (1999) compiled it by a compiler of language C++, and found that the same values of the complete elliptic integrals of the first and the second kind are obtained as those obtained by the procedure given by Morita (1978) when the argument of k', $\arg k'$, is between $-\pi/2$ and $\pi/2$, but some modifications are needed for the program in order to reproduce the results of Morita (1978) in the whole Riemann sheet where $-\pi < \arg k' < \pi$. The purpose of the present paper is to examine the convergence of a program obtained by modifying the one given in the book by Press et al. (1992).

We describe Carlson's algorithm in section 2, and a modified algorithm which is obtained as a result of the preceding paper, in section 3. A program of calculating the elliptic integrals of the first and the second kind and their analytic continuations with the aid of the modified algorithm, is given in the Appendix. In section 4, examination is given of the convergence of the program.

In the present paper, the square root of a complex number is always the principal value; when the square root of a complex number z is considered, the arguments of z and $z^{1/2}$ are in the ranges $(-\pi, \pi)$ and $(-\pi/2, \pi/2)$, respectively, so that the real part of a square root is always positive except for the real part of 0, that is zero.

2. Carlson's algorithm

The expressions used by Carlson (1979) for the elliptic integrals of the first and the second kind are

$$K(k) = R_F(0, (k')^2, 1), \quad F(\phi, k) = \sin\phi\, R_F(\cos^2\phi, (k'_\phi)^2, 1), \tag{1}$$

$$E(k) = R_F(0, (k')^2, 1) - \frac{1}{3}k^2 R_D(0, (k')^2, 1),$$

$$E(\phi, k) = \sin\phi\, R_F(\cos^2\phi, (k'_\phi)^2, 1) - \frac{1}{3}k^2\sin^3\phi\, R_D(\cos^2\phi, (k'_\phi)^2, 1), \tag{2}$$

where

$$k' = \sqrt{1 - k^2}, \quad k'_\phi = \sqrt{1 - k^2\sin^2\phi}, \tag{3}$$

$$R_F(x, y, z) = \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{(t + x)(t + y)(t + z)}}, \tag{4}$$

$$R_D(x, y, z) = \frac{3}{2}\int_0^\infty \frac{dt}{(t + z)\sqrt{(t + x)(t + y)(t + z)}}. \tag{5}$$

The algorithm which Carlson (1979) used in the calculation of $R_F(x, y, z)$ was based on the recursion formula:

$$R_F(x_n, y_n, z_n) = R_F(x_{n+1}, y_{n+1}, z_{n+1}), \tag{6}$$

and the expansion:

$$R_F(x_N, y_N, z_N) = \mu_N^{-1/2}\left[1 + \frac{1}{5}s_N^{(2)} + \frac{1}{7}s_N^{(3)} + \frac{1}{6}(s_N^{(2)})^2 + \frac{3}{11}s_N^{(2)}s_N^{(3)} + r_N\right], \tag{7}$$

where

$$x_{n+1} = (x_n + \lambda_n)/4, \quad y_{n+1} = (y_n + \lambda_n)/4, \quad z_{n+1} = (z_n + \lambda_n)/4, \tag{8}$$

$$\lambda_n = (x_n y_n)^{1/2} + (y_n z_n)^{1/2} + (z_n x_n)^{1/2}, \tag{9}$$

$$\mu_N = (x_N + y_N + z_N)/3, \tag{10}$$

$$X_N = 1 - x_N/\mu_N, \quad Y_N = 1 - y_N/\mu_N, \quad Z_N = 1 - z_N/\mu_N, \tag{11}$$

$$s_N^{(m)} = (X_N^m + Y_N^m + Z_N^m)/(2m), \quad m = 2, 3, \tag{12}$$

$$|r_N| < \frac{\varepsilon_N^6}{4(1 - \varepsilon_N)}, \quad \varepsilon_N = \max\{|X_N|, |Y_N|, |Z_N|\}. \tag{13}$$

After N iterations of (6), starting from $n = 0$, we obtain

$$R_F(x_0, y_0, z_0) = R_F(x_N, y_N, z_N). \tag{14}$$

We evaluate the righthand side by using (7) for N for which ε_N defined by (13) is sufficiently small.

The algorithm which Carlson (1979) used in the calculation of $R_D(x, y, z)$ was based on the recursion formula:

$$R_D(x_n, y_n, z_n) = \frac{3}{z_n^{1/2}(z_n + \lambda_n)} + \frac{1}{4}R_D(x_{n+1}, y_{n+1}, z_{n+1}), \tag{15}$$

and the expansion:

$$R_D(x_N, y_N, z_N) = \mu_N^{-3/2}\left[1 + \frac{3}{7}s_N^{(2)} + \frac{1}{3}s_N^{(3)} + \frac{3}{22}(s_N^{(2)})^2 + \frac{3}{11}s_N^{(4)} + \frac{3}{13}s_N^{(2)}s_N^{(3)} + \frac{3}{13}s_N^{(5)} + r_N\right], \tag{16}$$

where x_{n+1}, y_{n+1}, z_{n+1} and λ_n are given by (8) and (9), and

$$\mu_N = (x_N + y_N + 3z_N)/5, \tag{17}$$

$$X_N = 1 - x_N/\mu_N, \quad Y_N = 1 - y_N/\mu_N, \quad Z_N = 1 - z_N/\mu_N, \tag{18}$$

$$s_N^{(m)} = (X_N^m + Y_N^m + 3Z_N^m)/(2m), \quad m = 2, 3, 4, 5, \tag{19}$$

$$|r_N| < \frac{3\varepsilon_N^6}{(1 - \varepsilon_N)^{3/2}}, \quad \varepsilon_N = \max\{|X_N|, |Y_N|, |Z_N|\}. \tag{20}$$

After N iterations of (15), we obtain

$$R_D(x_0, y_0, z_0) = 3\sum_{n=0}^{N-1}\frac{4^{-n}}{z_n^{1/2}(z_n + \lambda_n)} + 4^{-N}R_D(x_N, y_N, z_N). \tag{21}$$

We evaluate the righthand side by using (16) for N for which ε_N defined by (20) is sufficiently small.

We can rewrite (8) with (9) as

$$\begin{aligned} x_{n+1} &= (x_n^{1/2} + y_n^{1/2})(x_n^{1/2} + z_n^{1/2})/4, \\ y_{n+1} &= (y_n^{1/2} + z_n^{1/2})(y_n^{1/2} + x_n^{1/2})/4, \\ z_{n+1} &= (z_n^{1/2} + x_n^{1/2})(z_n^{1/2} + y_n^{1/2})/4. \end{aligned} \tag{22}$$

This shows that, if x_n, y_n and z_n have arguments in the range $(-\pi, \pi)$, then x_{n+1}, y_{n+1} and z_{n+1} have arguments in the same range. This implies that x_n, y_n and z_n are analytic functions of x_0, y_0 and z_0 in the Riemann sheets satisfying

$$|\arg x_0| < \pi, \quad |\arg y_0| < \pi, \quad |\arg z_0| < \pi, \tag{23}$$

and hence the functions $R_F(x_0, y_0, z_0)$ and $R_D(x_0, y_0, z_0)$ calculated by this program are analytic functions of these variables x_0, y_0 and z_0 in these Riemann sheets.

As the result, $K(k)$ and $E(k)$ calculated by (1) and (2) with the aid of the algorithm are analytic in the Riemann sheet satisfying

$$-\pi/2 < \arg k' < \pi/2, \tag{24}$$

and $F(\phi, k)$ and $E(\phi, k)$ obtained by (1) and (2) are analytic in the Riemann sheets satisfying

$$-\pi/2 < \arg\cos\phi < \pi/2, \quad -\pi/2 < \arg k'_\phi = \arg\sqrt{1 - k^2\sin^2\phi} < \pi/2. \tag{25}$$

3. Modified algorithm

We now consider modified expressions:

$$K(k) = R_{FA}(0, k', 1), \quad F(\phi, k) = \sin\phi\, R_{FA}(\cos\phi, k'_\phi, 1), \tag{26}$$

$$E(k) = R_{FA}(0, k', 1) - \frac{1}{3}k^2 R_{DA}(0, k', 1),$$

$$E(\phi, k) = \sin\phi\, R_{FA}(\cos\phi, k'_\phi, 1) - \frac{1}{3}k^2\sin^3 R_{DA}(\cos\phi, k'_\phi, 1), \tag{27}$$

where

$$R_{FA}(x, y, z) = R_F(x^2, y^2, z^2), \quad R_{DA}(x, y, z) = R_D(x^2, y^2, z^2), \tag{28}$$

From equations (4) and (5), we obtain the homogeneities:

$$R_F(kx, ky, kz) = k^{-1/2}R_F(x, y, z), \quad R_D(kx, ky, kz) = k^{-3/2}R_D(x, y, z). \tag{29}$$

Using these in the recursion formulas (6) and (15) with (8), we obtain

$$R_F(x'_n, y'_n, 1) = \frac{2}{(1 + (x'_n)^{1/2})^{1/2}(1 + (y'_n)^{1/2})^{1/2}} R_F(x'_{n+1}, y'_{n+1}, 1), \tag{30}$$

$$R_D(x'_n, y'_n, 1) = \frac{3}{(1 + (x'_n)^{1/2})(1 + (y'_n)^{1/2})} + \frac{2}{(1 + (x'_n)^{1/2})^{3/2}(1 + (y'_n)^{1/2})^{3/2}} R_D(x'_{n+1}, y'_{n+1}, 1), \tag{31}$$

where

$$x'_{n+1} = \frac{(x'_n)^{1/2} + (y'_n)^{1/2}}{1 + (y'_n)^{1/2}}, \quad y'_{n+1} = \frac{(x'_n)^{1/2} + (y'_n)^{1/2}}{1 + (x'_n)^{1/2}}. \tag{32}$$

For the functions $R_{FA}(x, y, z)$ and $R_{DA}(x, y, z)$ defined by (28), the recursion formulas (30) and (31) with (32) are expressed as

$$R_{FA}(x'_n, y'_n, 1) = \frac{2}{(1 + x'_n)^{1/2}(1 + y'_n)^{1/2}} R_{FA}(x'_{n+1}, y'_{n+1}, 1), \tag{33}$$

$$R_{DA}(x'_n, y'_n, 1) = \frac{3}{(1 + x'_n)(1 + y'_n)} + \frac{2}{(1 + x'_n)^{3/2}(1 + y'_n)^{3/2}} R_{DA}(x'_{n+1}, y'_{n+1}, 1), \tag{34}$$

where

$$x'_{n+1} = \frac{(x'_n + y'_n)^{1/2}}{(1 + y'_n)^{1/2}}, \quad y'_{n+1} = \frac{(x'_n + y'_n)^{1/2}}{(1 + x'_n)^{1/2}}. \tag{35}$$

After N iterations, we obtain

$$R_{FA}(x'_0, y'_0, 1) = \left\{\prod_{n=0}^{N-1} \frac{2}{(1 + x'_n)^{1/2}(1 + y'_n)^{1/2}}\right\} R_{FA}(x'_N, y'_N, 1). \tag{36}$$

$$R_{DA}(x'_0, y'_0, 1) = \frac{3}{(1 + x'_0)(1 + y'_0)} + \sum_{n=1}^{N-1}\left\{\prod_{k=0}^{n-1} \frac{2}{(1 + x'_k)^{3/2}(1 + y'_k)^{3/2}}\right\} \frac{3}{(1 + x'_n)(1 + y'_n)}$$
$$+ \left\{\prod_{k=0}^{N-1} \frac{2}{(1 + x'_k)^{3/2}(1 + y'_k)^{3/2}}\right\} R_{DA}(x'_N, y'_N, 1). \tag{37}$$

The $R_{FA}(x'_N, y'_N, 1)$ and $R_{DA}(x'_N, y'_N, 1)$ on the righthand sides are evaluated with the aid of (28), (7) and (16) for N for which ε_N defined by (13) and (20) are both sufficiently small. We note that we can now evaluate the ε_N by

$$\varepsilon_N \simeq \max\{|1 - (x'_N)^2|, |1 - (y'_N)^2|\}, \tag{38}$$

when $\varepsilon_N \ll 1$.

Recursion relations (35) show that, if x'_n and y'_n have arguments in the range $(-\pi/2, \pi/2)$, the arguments of x'_{n+1} and y'_{n+1} are in the same range $(-\pi/2, \pi/2)$. This means that x'_n and y'_n for all positive integers $n \in \mathbf{N}$ have

arguments in the same range, and are analytic functions of x_0' and y_0' in the Riemann sheets satisfying

$$-\pi/2 < \arg x_0' < \pi/2, \quad -\pi/2 < \arg y_0' < \pi/2, \tag{39}$$

and that $R_{FA}(x_0', y_0', 1)$ and $R_{DA}(x_0', y_0', 1)$ calculated by the expressions (36) and (37) with (35) are analytic functions of x_0' and y_0' in the Riemann sheets satisfying (39). As the result, $K(k)$ and $E(k)$ calculated by (26) and (27) with the aid of the algorithm are analytic in the Riemann sheet satisfying (24), and $F(\phi, k)$ and $E(\phi, k)$ obtained by (26) and (27) are analytic in the Riemann sheet satisfying (25).

We now consider the case that x_0' is real and nonnegative, and

$$-\pi < \arg y_0' < \pi, \tag{40}$$

and then we see that x_1' and y_1' have arguments in the range $(-\pi/2, \pi/2)$, and hence x_n' and y_n' for all $n \geq 1$ have arguments in the range $(-\pi/2, \pi/2)$. This means that $R_{FA}(x_0', y_0', 1)$ and $R_{DA}(x_0', y_0', 1)$ calculated by the expressions (36) and (37) with (35) are analytic functions of y_0' in the range given by (40), when x_0' is real and nonnegative. As a result, $K(k)$ and $E(k)$ and their analytic continuations are calculated by (26) and (27) with the aid of the modified algorithm, in the Riemann sheet satisfying

$$-\pi < \arg k' < \pi, \tag{41}$$

and $F(\phi, k)$ and $E(\phi, k)$ and their analytic continuations are calculated by (26) and (27) in the Riemann sheet satisfying

$$-\pi < \arg k_\phi' < \pi \tag{42}$$

when $\cos\phi$ is real and nonnegative. This condition for $\cos\phi$ means either that ϕ is real and in the range $[-\pi/2, \pi/2]$, or that ϕ is pure imaginary.

4. Numerical calculation

Three programs of calculating the elliptic integrals of the first and the second kind, $K(k)$, $F(\phi, k)$, $E(k)$ and $E(\phi, k)$, and their analytic continuations, are given in the Appendix. The first one, which is called program C1 below, was constructed on the basis of the algorithm presented in the preceding section. In the program, the repetition of calculation is terminated when r_N given by (13) and (20) are both less than a required error ε, and hence $|\varepsilon_N| < \varepsilon^{1/6}/3^{1/6} \approx 0.83\varepsilon^{1/6}$ for ε_N estimated by (38). We take $\varepsilon = 10^{-6}$ or 10^{-12}.

In the second program, which is called program C2, we use

$$\begin{aligned} R_{FA}(x_N', y_N', 1) &= \left\{\frac{(x_N')^2 + (y_N')^2 + 1}{3}\right\}^{-1/2}\left[1 + \frac{1}{5}s_N^{(2)} + O(\varepsilon_N^3)\right] \\ &= \left[1 - \frac{1}{2}\left\{\frac{(x_N')^2 + (y_N')^2 + 1}{3} - 1\right\}\right](1 + r_N^{(2)}), \end{aligned} \tag{43}$$

in place of (28) and (7), where

$$|r_N^{(2)}| < \frac{1}{5}\varepsilon_N^2(1 + O(\varepsilon_N)), \tag{44}$$

and

$$\begin{aligned} R_{DA}(x_N', y_N', 1) &= \left\{\frac{(x_N')^2 + (y_N')^2 + 3}{5}\right\}^{-3/2}\left[1 + \frac{3}{7}s_N^{(2)} + O(\varepsilon_N^3)\right] \\ &= \left[1 - \frac{3}{2}\left\{\frac{(x_N')^2 + (y_N')^2 + 3}{5} - 1\right\}\right](1 + r_N^{(2)}), \end{aligned} \tag{45}$$

in place of (28) and (16), where

$$|r_N^{(2)}| < \frac{69}{175}(1 + O(\varepsilon_N)). \tag{46}$$

In the program C2, the repetition of calculation is terminated when $r_N^{(2)}$ given by (44) and (46) are both less than a required error ε, and hence $|\varepsilon_N| < \sqrt{2.5}\varepsilon^{1/2} \approx 1.58\varepsilon^{1/2}$.

When we use the program C1 or C2 in calculating the complete integrals $K(k)$ and $E(k)$, we can answer 0 or 1 meaning yes or no to the question "complete integral?", but when $|k'| \leq 10^{-10}$, we have to answer 0 meaning yes and give the value k', not k, in order to get the correct values. If we answer 1 meaning no and give the values of k' and $\phi = \pi/2$, we get results with no significant figures, in the calculations $k = (1 - k'^2)^{1/2}$ and $k' = (1 - k^2)^{1/2}$.

The third program which we call A, is the one based on the algorithm given by Morita (1978), in which $K(k)$ and $E(k)$ are calculated by the arithmetic-geometric mean procedure. In program A, the calculation is terminated when the absolute value of the difference of two successive values obtained, divided by one of these values, is less than ε, so that the obtained value has a relative error less than ε.

Numererical calculations of $F(\phi, k)$ and $E(\phi, k)$ were performed for real ϕ and complex k', when

$$|k'| = 10^{\pm 0.01}, 10^{\pm 0.1}, 10^{\pm 1}, 10^{\pm 10}, 10^{\pm 100},$$

$$\arg k' = 0, \pm\pi/6, \pm\pi/3, \pm\pi/2, \pm 2\pi/3, \pm 5\pi/6, \pm\pi \mp 10^{-6},$$

and $\varepsilon = 10^{-6}$ or 10^{-12}. The value of $N - 1$ when the calculation is terminated is listed in Table 1, for the case of $\phi = \pi/2$, and hence for the calculations of $K(k)$ and $E(k)$. Two numbers for programs C1 and A are listed for each value of k', when $\log_{10}|k'| > 0$ and $\arg k' \geq 0$. We note that the first number is slightly smaller or equal to the second number. The numbers for program C2, which are not listed here, are greater than those for C1 by

Table 1. Number of repetitions, $N - 1$, in the calculation with relative error $\varepsilon = 10^{-6}$ or 10^{-12}.

(a) $\varepsilon = 10^{-6}$

$\lvert k'\rvert$	$10^{0.01}$		$10^{0.1}$		10		10^{10}		10^{100}	
$\arg k'$	C1	A	C1	A	C1	A	C1	A	C1	A
0	2	2	2	3	3	4	6	7	9	10
$\pi/6$	2	3	2	3	3	4	6	7	9	10
$\pi/3$	2	4	2	4	3	4	6	7	9	10
$\pi/2$	3	4	3	4	3	5	6	7	9	10
$2\pi/3$	3	5	3	5	3	5	6	7	9	10
$5\pi/6$	3	5	3	5	4	5	6	7	9	10
π	4	6	3	5	4	5	6	7	9	10

(b) $\varepsilon = 10^{-12}$

$\lvert k'\rvert$	$10^{0.01}$		$10^{0.1}$		10		10^{10}		10^{100}	
$\arg k'$	C1	A	C1	A	C1	A	C1	A	C1	A
0	4	3	3	4	5	5	8	8	11	11
$\pi/6$	4	4	4	4	5	5	8	8	11	11
$\pi/3$	4	5	4	5	5	5	8	8	11	11
$\pi/2$	4	5	4	5	5	5	8	8	11	11
$2\pi/3$	4	5	5	5	5	6	8	8	11	11
$5\pi/6$	4	6	4	6	5	6	8	8	11	11
π	6	7	5	6	5	6	8	8	11	11

Table 2. CPU time in seconds for 100,000 repetitions of the calculation with relative error $\varepsilon = 10^{-6}$ or 10^{-12}.

(a) $\varepsilon = 10^{-6}$

$\lvert k'\rvert$	$10^{0.01}$		$10^{0.1}$		10		10^{10}		10^{100}	
$\arg k'$	C1	A	C1	A	C1	A	C1	A	C1	A
0	8	2	8	3	9	5	14	6	19	9
$\pi/6$	10	5	10	5	13	6	20	11	25	14
$\pi/3$	11	6	10	7	14	6	20	10	26	13
$\pi/2$	13	6	13	6	13	8	20	10	25	14
$2\pi/3$	13	8	13	8	13	8	20	10	26	14
$5\pi/6$	13	8	13	8	16	7	20	10	26	14
π	16	9	13	8	16	7	19	10	26	13

(b) $\varepsilon = 10^{-12}$

$\lvert k'\rvert$	$10^{0.01}$		$10^{0.1}$		10		10^{10}		10^{100}	
$\arg k'$	C1	A	C1	A	C1	A	C1	A	C1	A
0	12	3	10	5	13	6	18	9	23	12
$\pi/6$	16	7	16	7	19	8	25	13	31	17
$\pi/3$	16	9	16	9	19	8	25	13	31	16
$\pi/2$	16	8	15	8	18	9	24	13	31	16
$2\pi/3$	16	9	19	8	19	10	25	13	31	17
$5\pi/6$	16	10	15	11	18	10	25	13	31	17
π	20	11	19	10	19	9	24	13	31	17

$2 \sim 3$ when $\varepsilon = 10^{-6}$ and by $6 \sim 7$ when $\varepsilon = 10^{-12}$. In the present programs, the integrals of the first and second kinds are calculated at the same time. When we do calculations separately and evaluate the N when we get the results in the required accuracy, the same numbers of N are obtained for almost all the cases. We confirm that the listed numbers in the tables do not depend on the signs of $\log_{10}|k'|$ and $\arg k'$. In confirming this for the programs C1 and C2, we have had to answer 0 to "complete integral?" when $|k'| < 10^{-10}$, as stated above.

In Table 2, we compare the CPU times required in the calculations. The numbers listed are in seconds when the same calculations are repeated 100,000 times by the same computer. The two numbers for each value of k' were taken in the same way as in Table 1. We see that the first number is almost twice of the second number. The

```
// CC ell/iise12b.cc -lm -lcomplex
// g++ ell/iise12b.cc
#include <stdio.h>
#include <math.h>
#include <complex.h>
#define N 12 // 6
#define complex double_complex
#define sqrtc sqrt
#define TINY 3.0e-138 // #define TINY 3.0e-38
#define BIG 3.0e137 // #define BIG 3.0e37
#define C1(1.0/24.0)
#define C2 0.1
#define C3(3.0/44.0)
#define C4(1.0/14.0)
#define TINYd 1.0e-125 // #define TINYd 1.0e-25
#define BIGd 4.5e121 // #define BIGd 4.5e21
#define C1d(3.0/14.0)
#define C2d(1.0/6.0)
#define C3d(9.0/22.0)
#define C4d(3.0/26.0)
#define C5d(0.25*C3d)
#define C6d(1.5*C4d)
#define pi 3.1415926535897932
#define pihf(0.5*pi)
void ell12(complex phi, complex ak, complex ck, int sgnck); complex el1,el2;
void rfrd(complex xrt, complex yrt); complex rf,rd;
void ell12b(complex phi, complex ak, complex ck, int sgnck);
void rfrdb(complex xrt, complex yrt);
// complex sqrtc(complex z);
void cel12(complex ck); complex cel1,cel2; int nrep;
double RERR,ERRTOL,ERRTOLb;
main(){
    double rp,ip; complex ck,ak,phi; int sgnck,choicekkc,ans;
    if(N==12){RERR= 1.0e-12; ERRTOL=0.008; ERRTOLb=0.0000015;}
    if(N==6){RERR= 1.0e-6; ERRTOL=0.08; ERRTOLb=0.0015;}
    printf(" k or kc? if(k) answer 0; if(kc) answer 1! \n choicekc = ");
    scanf("%d", &choicekkc);
    if(choicekkc){printf(" real(kc) imag(kc) = "); scanf("%lf%lf",&rp,&ip);
        ck=complex(rp,ip); ak=sqrt(1-ck*ck);
        if(rp<0) sgnck= -1;
        else sgnck=1;}
    else{printf(" real(k) imag(k) = "); scanf("%lf%lf",&rp,&ip);
        ak=complex(rp,ip); ck=sqrt(1-ak*ak);
        printf(" fn or anal. conti.? if(fn) answer 0; if(anal. conti.) answer 1! \n");
        printf(" choiceac = "); scanf("%d",&sgnck); sgnck=1-2*sgnck;
        if(sgnck<0) ck= -ck;}
    printf(" complete integral? if(yes) answer 0; if(no) answer 1! \n");
    printf(" answer = "); scanf("%d",&ans);
    if(ans){printf(" real(phi)/pihf imag(phi) = "); scanf("%lf%lf",&rp,&ip);}
    else{rp=1; ip=0;}
    phi=complex(rp*pihf,ip);
    cel12(ck);
    printf(" ck = %f %f \n",real(ck),imag(ck));
    printf(" cel1 = %f %f cel2 = %f %f \n",real(cel1),imag(cel1),real(cel2),imag(cel2));
    printf(" ak = %f %f phi = %f %f ",real(ak),imag(ak),real(phi),imag(phi));
    printf(" sgnck = %d \n",sgnck);
    if(ans==0) sgnck=0;
    ell12(phi,ak,ck,sgnck);
    printf(" el1 = %f %f el2 = %f %f \n",real(el1),imag(el1),real(el2),imag(el2));
    ell12b(phi,ak,ck,sgnck);
    printf(" el1 = %f %f el2 = %f %f \n",real(el1),imag(el1),real(el2),imag(el2));}
```

Fig. 1 C++ program of calulating the elliptic integrals of the first and the second kind with complex modulus (to continue to Fig. 2).

numbers for program C2, which are not listed here, are greater than those for C1 by 2 ~ 5 when $\varepsilon = 10^{-6}$ and by 5 ~ 10 when $\varepsilon = 10^{-12}$, so that the CPU time required by program C2 is less than twice of the time required by C1.

When the calculation is done in the arithmetic-geometric mean procedure, the error ε'_N when we terminate the calculation at the Nth repetition, behaves as $\varepsilon'_N \approx (\varepsilon'_{N-1})^2/8$; see Morita and Horiguchi (1973). This means that, when we terminate the calculation by the error of 10^{-6} or 10^{-12} in Tables 1 and 2, the obtained values of elliptic integrals already have only an error of 10^{-12} or 10^{-24}. When we use the program C1 and C2, it was found that the obtained results have relative error less than 5×10^{-10} or 10^{-14} and 3×10^{-7} or 3×10^{-13}, respectively. The only merit of program C2 over C1 is that the program is simpler.

The present result shows that the calculation in the arithmetic-geometric mean procedure, is superior to the calculation based on Carlson's algorithm if we calculate only the complete elliptic integrals $K(k)$ and $E(k)$.

```
161 void ell12(complex phi, complex ak, complex ck, int sgnck){complex c,s,q,qrt,sk2;
162     if(sgnck==0){c=0; qrt=ck;}
163     else{s=sin(phi); c=cos(phi); sk2=s*ak; sk2=sk2*sk2; q=1-sk2; qrt=sqrtc(q);
164         if(sgnck<0) qrt= -qrt;}
165     rfrd(c,qrt); el1=s*rf; el2=el1-s*sk2*rd/3;}
166 void rfrd(complex xrt, complex yrt){
167     complex avi,delx,dely,delz,e2,e3,fac;
168     complex avid,delxd,delyd,delzd,ea,eb,ec,ed,ee,sumd,facd;
169     complex alamb,sqrtx,sqrty,sqrtzi,xt,yt,zi;
170     sqrtx=xrt; sqrty=yrt; xt=xrt*xrt; yt=yrt*yrt;
171     // if(abs(xt+yt)TINY || abs(xt+1)<TINY || abs(yt+1)<TINY || abs(xt)>BIG || abs(yt)>BIG))
172     // printf(" invalid arguments in rf \n");
173     // if(abs(xt+yt)<TINYd || abs(xt)>BIGd || abs(yt)>BIGd))
174     // printf(" invalid arguments in rd \n");
175     fac=1; sumd=0; facd=1; nrep=0;
176     while(1){alamb=sqrtx*sqrty+sqrtx+sqrty; zi=1/(1+alamb);
177         xt=xt+alamb; yt=yt+alamb; xt=xt*zi; yt=yt*zi; sqrtzi=2*sqrtc(zi);
178         fac=fac*sqrtzi; sumd=sumd+facd*zi; facd=facd*zi*sqrtzi;
179         delx=1-xt; dely=1-yt;
180         if(abs(delx)<ERRTOL && abs(dely)<ERRTOL)break;
181         sqrtx=sqrtc(xt); sqrty=sqrtc(yt); nrep++;}
182     avi=3/(xt+yt+1); delx=1-xt*avi; dely=1-yt*avi; delz=1-avi;
183     e2=delx*dely-delz*delz; e3=delx*dely*delz;
184     rf=fac*(1+(C1*e2-C2-C3*e3)*e2+C4*e3)*sqrtc(avi);
185     avid=5/(xt+yt+3); delxd=1-xt*avid; delyd=1-yt*avid; delzd=1.0-avid;
186     ea=delxd*delyd; eb=delzd*delzd; ec=ea-eb; ed=ea-6.0*eb; ee=ed+ec+ec;
187     rd=3*sumd+facd*(1+ed*(-C1d+C5d*ed-C6d*delz*ee)
188     +delzd*(C2d*ee+delzd*(-C3d*ec+delzd*C4d*ea)))*avid*sqrtc(avid);}
189 void ell12b(complex phi, complex ak, complex ck, int sgnck){complex c,s,q,qrt,sk2;
190     if(sgnck==0){c=0; qrt=ck;}
191     else{s=sin(phi); c=cos(phi); sk2=s*ak; sk2=sk2*sk2; q=1-sk2; qrt=sqrtc(q);
192         if(sgnck<0) qrt= -qrt;}
193     rfrdb(c,qrt); el1=s*rf; el2=el1-s*sk2*rd/3;}
194 void rfrdb(complex xrt, complex yrt){
195     complex delx,dely,delz,fac,sumd,facd;
196     complex alamb,sqrtx,sqrty,sqrtzi,xt,yt,xyt,zi;
197     sqrtx=xrt; sqrty=yrt; xt=xrt*xrt; yt=yrt*yrt;
198     // if(abs(xt+yt)TINY || abs(xt+1)<TINY || abs(yt+1)<TINY || abs(xt)>BIG || abs(yt)>BIG))
199     // printf(" invalid arguments in rf \n");
200     // if(abs(xt+yt)<TINYd || abs(xt)>BIGd || abs(yt)>BIGd))
201     // printf(" invalid arguments in rd \n");
202     fac=1; sumd=0; facd=1; nrep=0;
203     while(1){alamb=sqrtx*sqrty+sqrtx+sqrty; zi=1/(1+alamb);
204         xt=xt+alamb; yt=yt+alamb; xt=xt*zi; yt=yt*zi; sqrtzi=2*sqrtc(zi);
205         fac=fac*sqrtzi; sumd=sumd+facd*zi; facd=facd*zi*sqrtzi;
206         delx=1-xt; dely=1-yt;
207         if(abs(delx)<ERRTOLb && abs(dely)<ERRTOLb) break;
208         sqrtx=sqrtc(xt); sqrty=sqrtc(yt); nrep++;}
209     xyt=xt+yt; rf=fac*(8-xyt)/6.0; rd=3*sumd+facd*(1.6-0.3*xyt);}
210 void cel12(complex ck){complex kc,ai,sa,aa,sb,ta;
211     // if(abs(ck)<TINY){cel1=0; cel2=0; return;}
212     kc=ck; ta=ck*ck; ai=1; sb=1; nrep=0;
213     while(1){aa=2/(1+kc); ai=ai*aa; sa=(sb+ta)*aa/2;
214         if(abs(1.0-aa)<RERR && abs(1.0-sa/sb)<RERR) break;
215         ta=(ta+kc*sb)*aa*aa/2; sb=sa; kc=sqrtc(kc)*aa; nrep++;}
216     cel1=pihf*ai; cel2=pihf*sa;}
217 // complex sqrtc(complex z){ double a, th, zsqr,zsqi; complex zsq;
218 // a=abs(z); a=sqrt(a); th=arg(z); th *=0.5;
219 // zsqr=a*cos(th);zsqi=a*sin(th); zsq=complex(zsqr,zsqi); return zsq;}
```

Fig. 2 Function procedures of calculating $F(\phi, k)$, $E(\phi, k)$, $R_{FA}(x, y, 1)$, $R_{DA}(x, y, 1)$, $K(k)$ and $E(k)$.

When we have to calculate the incomplete integrals $F(\phi, k)$ and $E(\phi, k)$ for $\phi \neq \pi/2$ and complex k', we have only the procedure based on Carlson's algorithm, and hence we have to use it.

Appendix: Program in C++

A program of calculating the elliptic integrals of the first and the second kind, $K(k)$, $F(\phi, k)$, $E(k)$ and $E(\phi, k)$, and their analytic continuations, is given in Figs. 1 and 2. We assume that two complex numbers, one for modulus k or complementary modulus k' and the other for ϕ, are given to the program. The program is written in language C++ and is compiled as indicated in row 100 or 101 in Fig. 1. Figure 1 shows definitions of numerical constants, declarations of functions and the main program. The numerical constants are mostly taken from the book of Press et al. (1992). In Fig. 2, two functions to calculate the elliptic integrals $F(\phi, k)$ and $E(\phi, k)$ are given in rows 161 ~ 165 and in rows 189 ~ 193. The values of R_{FA} and R_{DA} are calculated by using the algorithms C1 and C2 in the functions given in rows 166 ~ 188 and in rows 194 ~ 209, respectively. The programs for R_{FA} and R_{DA} are written by modifying the program for functions R_F and R_D given in the book of Press et al. (1992). In Fig. 2, the part of rows 210 ~ 216 is the program of calculating the complete elliptic integrals of the first and the second kind and their analytic continuations, taken from the paper of Morita (1978).

The program given is the form when we compile it by using a g++ compiler as indicated in row 101. When we compile it by using a C++ compiler as indicated in row 100, we have to delete row 106.

In rows 217 ~ 219, function *sqrtc*(z) is defined, which gives the square root of a complex number z. The function *sqrt*(z) prepared in the C++ compiler in the computer used could give only 8 valid digits for z which has an argument nearly equal to π, that was not satisfactory in the present calculation. The function *sqrtc*(z) could give the square root in enough accuracy. When we use the function *sqrtc*(z), we have to delete row 107 and also delete "//" in rows 128, 217, 218 and 219.

REFERENCES

[1] Bulirsch, R., (1965), Handbook series special functions, numerical calculation of elliptic integrals and elliptic functions, Numer. Math. **7**, 78–90.
[2] Carlson, B. C., (1979), Computing Elliptic Integrals by Duplication, Numer. Math. **33**, 1–16.
[3] Morita, T., (1978), Calculation of the Complete Elliptic Integrals with Complex Modulus, Numer. Math. **29**, 233–236.
[4] Morita, T., (1999), Calculation of the elliptic integrals of the first and second kinds with complex modulus, Numer. Math. **82**, 677–688.
[5] Morita, T. and Horiguchi, T., (1973), Convergence of the arithmetic-geometric mean procedure for the complex variables and the calculation of the complete elliptic integrals with complex modulus, Numer. Math. **20**, 425–430.
[6] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., (1992), Numerical Recipes in C, The Art of Scientific Computing, Second Edition, pp. 261–269, Cambridge U.P., Cambridge, New York, Melbourne.