Other methods for computing Dawson's integral are also known [2,3].

CITED REFERENCES AND FURTHER READING:

Rybicki, G.B. 1989, *Computers in Physics*, vol. 3, no. 2, pp. 85–87. [1]

Cody, W.J., Pociorek, K.A., and Thatcher, H.C. 1970, *Mathematics of Computation*, vol. 24, pp. 171–178. [2]

McCabe, J.H. 1974, *Mathematics of Computation*, vol. 28, pp. 811–816. [3]

6.11 Elliptic Integrals and Jacobian Elliptic Functions

Elliptic integrals occur in many applications, because any integral of the form

$$\int R(t, s)\, dt \tag{6.11.1}$$

where R is a rational function of t and s, and s is the square root of a cubic or quartic polynomial in t, can be evaluated in terms of elliptic integrals. Standard references [1] describe how to carry out the reduction, which was originally done by Legendre. Legendre showed that only three basic elliptic integrals are required. The simplest of these is

$$I_1 = \int_y^x \frac{dt}{\sqrt{(a_1 + b_1 t)(a_2 + b_2 t)(a_3 + b_3 t)(a_4 + b_4 t)}} \tag{6.11.2}$$

where we have written the quartic s^2 in factored form. In standard integral tables [2], one of the limits of integration is always a zero of the quartic, while the other limit lies closer than the next zero, so that there is no singularity within the interval. To evaluate I_1, we simply break the interval $[y, x]$ into subintervals, each of which either begins or ends on a singularity. The tables, therefore, need only distinguish the eight cases in which each of the four zeros (ordered according to size) appears as the upper or lower limit of integration. In addition, when one of the b's in (6.11.2) tends to zero, the quartic reduces to a cubic, with the largest or smallest singularity moving to $\pm\infty$; this leads to eight more cases (actually just special cases of the first eight). The sixteen cases in total are then usually tabulated in terms of Legendre's standard elliptic integral of the 1st kind, which we will define below. By a change of the variable of integration t, the zeros of the quartic are mapped to standard locations on the real axis. Then only two dimensionless parameters are needed to tabulate Legendre's integral. However, the symmetry of the original integral (6.11.2) under permutation of the roots is concealed in Legendre's notation. We will get back to Legendre's notation below. But first, here is a better way:

Carlson [3] has given a new definition of a standard elliptic integral of the first kind,

$$R_F(x, y, z) = \frac{1}{2}\int_0^\infty \frac{dt}{\sqrt{(t+x)(t+y)(t+z)}} \tag{6.11.3}$$

where x, y, and z are nonnegative and at most one is zero. By standardizing the range of integration, he retains permutation symmetry for the zeros. (Weierstrass' canonical form also has this property.) Carlson first shows that when x or y is a zero of the quartic in (6.11.2), the integral I_1 can be written in terms of R_F in a form that is symmetric under permutation of the *remaining* three zeros. In the general case when neither x nor y is a zero, two such R_F functions can be combined into a single one by an *addition theorem*, leading to the fundamental formula

$$I_1 = 2R_F(U_{12}^2, U_{13}^2, U_{14}^2) \tag{6.11.4}$$

where

$$U_{ij} = (X_i X_j Y_k Y_m + Y_i Y_j X_k X_m)/(x - y) \tag{6.11.5}$$

$$X_i = (a_i + b_i x)^{1/2}, \qquad Y_i = (a_i + b_i y)^{1/2} \tag{6.11.6}$$

and i, j, k, m is any permutation of $1, 2, 3, 4$. A short-cut in evaluating these expressions is

$$\begin{aligned} U_{13}^2 &= U_{12}^2 - (a_1 b_4 - a_4 b_1)(a_2 b_3 - a_3 b_2) \\ U_{14}^2 &= U_{12}^2 - (a_1 b_3 - a_3 b_1)(a_2 b_4 - a_4 b_2) \end{aligned} \tag{6.11.7}$$

The U's correspond to the three ways of pairing the four zeros, and I_1 is thus manifestly symmetric under permutation of the zeros. Equation (6.11.4) therefore reproduces all sixteen cases when one limit is a zero, and also includes the cases when neither limit is a zero.

Thus Carlson's function allows arbitrary ranges of integration and arbitrary positions of the branch points of the integrand relative to the interval of integration. To handle elliptic integrals of the second and third kind, Carlson defines the standard integral of the third kind as

$$R_J(x, y, z, p) = \frac{3}{2} \int_0^\infty \frac{dt}{(t + p)\sqrt{(t + x)(t + y)(t + z)}} \tag{6.11.8}$$

which is symmetric in x, y, and z. The degenerate case when two arguments are equal is denoted

$$R_D(x, y, z) = R_J(x, y, z, z) \tag{6.11.9}$$

and is symmetric in x and y. The function R_D replaces Legendre's integral of the second kind. The degenerate form of R_F is denoted

$$R_C(x, y) = R_F(x, y, y) \tag{6.11.10}$$

It embraces logarithmic, inverse circular, and inverse hyperbolic functions.

Carlson [4-7] gives integral tables in terms of the exponents of the linear factors of the quartic in (6.11.1). For example, the integral where the exponents are $(\frac{1}{2}, \frac{1}{2}, -\frac{1}{2}, -\frac{3}{2})$ can be expressed as a single integral in terms of R_D; it accounts for 144 separate cases in Gradshteyn and Ryzhik [2]!

Refer to Carlson's papers [3-7] for some of the practical details in reducing elliptic integrals to his standard forms, such as handling complex conjugate zeros.

Turn now to the numerical evaluation of elliptic integrals. The traditional methods [8] are Gauss or Landen transformations. *Descending* transformations decrease the modulus k of the Legendre integrals towards zero, *increasing* transformations increase it towards unity. In these limits the functions have simple analytic expressions. While these methods converge quadratically and are quite satisfactory for integrals of the first and second kinds, they generally lead to loss of significant figures in certain regimes for integrals of the third kind. Carlson's algorithms [9,10], by contrast, provide a unified method for all three kinds with no significant cancellations.

The key ingredient in these algorithms is the *duplication theorem*:

$$\begin{aligned} R_F(x, y, z) &= 2R_F(x + \lambda, y + \lambda, z + \lambda) \\ &= R_F\left(\frac{x + \lambda}{4}, \frac{y + \lambda}{4}, \frac{z + \lambda}{4}\right) \end{aligned} \tag{6.11.11}$$

where

$$\lambda = (xy)^{1/2} + (xz)^{1/2} + (yz)^{1/2} \tag{6.11.12}$$

This theorem can be proved by a simple change of variable of integration [11]. Equation (6.11.11) is iterated until the arguments of R_F are nearly equal. For equal arguments we have

$$R_F(x, x, x) = x^{-1/2} \tag{6.11.13}$$

When the arguments are close enough, the function is evaluated from a fixed Taylor expansion about (6.11.13) through fifth-order terms. While the iterative part of the algorithm is only linearly convergent, the error ultimately decreases by a factor of $4^6 = 4096$ for each iteration. Typically only two or three iterations are required, perhaps six or seven if the initial values of the arguments have huge ratios. We list the algorithm for R_F here, and refer you to Carlson's paper [9] for the other cases.

Stage 1: For $n = 0, 1, 2, \ldots$ compute

$$\begin{aligned}
&\mu_n = (x_n + y_n + z_n)/3 \\
&X_n = 1 - (x_n/\mu_n), \quad Y_n = 1 - (y_n/\mu_n), \quad Z_n = 1 - (z_n/\mu_n) \\
&\epsilon_n = \max(|X_n|, |Y_n|, |Z_n|)
\end{aligned}$$

If $\epsilon_n < \text{tol}$ go to Stage 2; else compute

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

and repeat this stage.

Stage 2: Compute

$$\begin{aligned}
&E_2 = X_n Y_n - Z_n^2, \quad E_3 = X_n Y_n Z_n \\
&R_F = (1 - \tfrac{1}{10}E_2 + \tfrac{1}{14}E_3 + \tfrac{1}{24}E_2^2 - \tfrac{3}{44}E_2 E_3)/(\mu_n)^{1/2}
\end{aligned}$$

In some applications the argument p in R_J or the argument y in R_C is negative, and the Cauchy principal value of the integral is required. This is easily handled by using the formulas

$$\begin{aligned}
&R_J(x, y, z, p) = \\
&\qquad [(\gamma - y)R_J(x, y, z, \gamma) - 3R_F(x, y, z) + 3R_C(xz/y, p\gamma/y)] \,/(y - p)
\end{aligned} \tag{6.11.14}$$

where

$$\gamma \equiv y + \frac{(z - y)(y - x)}{y - p} \tag{6.11.15}$$

is positive if p is negative, and

$$R_C(x, y) = \left(\frac{x}{x - y}\right)^{1/2} R_C(x - y, -y) \tag{6.11.16}$$

The Cauchy principal value of R_J has a zero at some value of $p < 0$, so (6.11.14) will give some loss of significant figures near the zero.

```
#include <math.h>
#include "nrutil.h"
#define ERRTOL 0.08
#define TINY 1.5e-38
#define BIG 3.0e37
#define THIRD (1.0/3.0)
#define C1 (1.0/24.0)
#define C2 0.1
#define C3 (3.0/44.0)
#define C4 (1.0/14.0)

float rf(float x, float y, float z)
```
Computes Carlson's elliptic integral of the first kind, $R_F(x, y, z)$. x, y, and z must be nonnegative, and at most one can be zero. TINY must be at least 5 times the machine underflow limit, BIG at most one fifth the machine overflow limit.
```
{
    float alamb,ave,delx,dely,delz,e2,e3,sqrtx,sqrty,sqrtz,xt,yt,zt;

    if (FMIN(FMIN(x,y),z) < 0.0 || FMIN(FMIN(x+y,x+z),y+z) < TINY ||
        FMAX(FMAX(x,y),z) > BIG)
            nrerror("invalid arguments in rf");
    xt=x;
    yt=y;
    zt=z;
    do {
        sqrtx=sqrt(xt);
        sqrty=sqrt(yt);
        sqrtz=sqrt(zt);
        alamb=sqrtx*(sqrty+sqrtz)+sqrty*sqrtz;
        xt=0.25*(xt+alamb);
        yt=0.25*(yt+alamb);
        zt=0.25*(zt+alamb);
        ave=THIRD*(xt+yt+zt);
        delx=(ave-xt)/ave;
        dely=(ave-yt)/ave;
        delz=(ave-zt)/ave;
    } while (FMAX(FMAX(fabs(delx),fabs(dely)),fabs(delz)) > ERRTOL);
    e2=delx*dely-delz*delz;
    e3=delx*dely*delz;
    return (1.0+(C1*e2-C2-C3*e3)*e2+C4*e3)/sqrt(ave);
}
```

A value of 0.08 for the error tolerance parameter is adequate for single precision (7 significant digits). Since the error scales as ϵ_n^6, we see that 0.0025 will yield double precision (16 significant digits) and require at most two or three more iterations. Since the coefficients of the sixth-order truncation error are different for the other elliptic functions, these values for the error tolerance should be changed to 0.04 and 0.0012 in the algorithm for R_C, and 0.05 and 0.0015 for R_J and R_D. As well as being an algorithm in its own right for certain combinations of elementary functions, the algorithm for R_C is used repeatedly in the computation of R_J.

The C implementations test the input arguments against two machine-dependent constants, TINY and BIG, to ensure that there will be no underflow or overflow during the computation. We have chosen conservative values, corresponding to a machine minimum of 3×10^{-39} and a machine maximum of 1.7×10^{38}. You can always extend the range of admissible argument values by using the homogeneity relations (6.11.22), below.

```
#include <math.h>
#include "nrutil.h"
#define ERRTOL 0.05
```

```
#define TINY 1.0e-25
#define BIG 4.5e21
#define C1 (3.0/14.0)
#define C2 (1.0/6.0)
#define C3 (9.0/22.0)
#define C4 (3.0/26.0)
#define C5 (0.25*C3)
#define C6 (1.5*C4)

float rd(float x, float y, float z)
```

Computes Carlson's elliptic integral of the second kind, $R_D(x,y,z)$. x and y must be nonnegative, and at most one can be zero. z must be positive. TINY must be at least twice the negative 2/3 power of the machine overflow limit. BIG must be at most $0.1 \times$ ERRTOL times the negative 2/3 power of the machine underflow limit.

```
{
    float alamb,ave,delx,dely,delz,ea,eb,ec,ed,ee,fac,sqrtx,sqrty,
        sqrtz,sum,xt,yt,zt;

    if (FMIN(x,y) < 0.0 || FMIN(x+y,z) < TINY || FMAX(FMAX(x,y),z) > BIG)
        nrerror("invalid arguments in rd");
    xt=x;
    yt=y;
    zt=z;
    sum=0.0;
    fac=1.0;
    do {
        sqrtx=sqrt(xt);
        sqrty=sqrt(yt);
        sqrtz=sqrt(zt);
        alamb=sqrtx*(sqrty+sqrtz)+sqrty*sqrtz;
        sum += fac/(sqrtz*(zt+alamb));
        fac=0.25*fac;
        xt=0.25*(xt+alamb);
        yt=0.25*(yt+alamb);
        zt=0.25*(zt+alamb);
        ave=0.2*(xt+yt+3.0*zt);
        delx=(ave-xt)/ave;
        dely=(ave-yt)/ave;
        delz=(ave-zt)/ave;
    } while (FMAX(FMAX(fabs(delx),fabs(dely)),fabs(delz)) > ERRTOL);
    ea=delx*dely;
    eb=delz*delz;
    ec=ea-eb;
    ed=ea-6.0*eb;
    ee=ed+ec+ec;
    return 3.0*sum+fac*(1.0+ed*(-C1+C5*ed-C6*delz*ee)
        +delz*(C2*ee+delz*(-C3*ec+delz*C4*ea)))/(ave*sqrt(ave));
}
```

```
#include <math.h>
#include "nrutil.h"
#define ERRTOL 0.05
#define TINY 2.5e-13
#define BIG 9.0e11
#define C1 (3.0/14.0)
#define C2 (1.0/3.0)
#define C3 (3.0/22.0)
#define C4 (3.0/26.0)
#define C5 (0.75*C3)
#define C6 (1.5*C4)
#define C7 (0.5*C2)
#define C8 (C3+C3)
```

```
float rj(float x, float y, float z, float p)
```
Computes Carlson's elliptic integral of the third kind, $R_J(x,y,z,p)$. x, y, and z must be nonnegative, and at most one can be zero. p must be nonzero. If $p < 0$, the Cauchy principal value is returned. TINY must be at least twice the cube root of the machine underflow limit, BIG at most one fifth the cube root of the machine overflow limit.
```
{
    float rc(float x, float y);
    float rf(float x, float y, float z);
    float a,alamb,alpha,ans,ave,b,beta,delp,delx,dely,delz,ea,eb,ec,
        ed,ee,fac,pt,rcx,rho,sqrtx,sqrty,sqrtz,sum,tau,xt,yt,zt;

    if (FMIN(FMIN(x,y),z) < 0.0 || FMIN(FMIN(x+y,x+z),FMIN(y+z,fabs(p))) < TINY
        || FMAX(FMAX(x,y),FMAX(z,fabs(p))) > BIG)
            nrerror("invalid arguments in rj");
    sum=0.0;
    fac=1.0;
    if (p > 0.0) {
        xt=x;
        yt=y;
        zt=z;
        pt=p;
    } else {
        xt=FMIN(FMIN(x,y),z);
        zt=FMAX(FMAX(x,y),z);
        yt=x+y+z-xt-zt;
        a=1.0/(yt-p);
        b=a*(zt-yt)*(yt-xt);
        pt=yt+b;
        rho=xt*zt/yt;
        tau=p*pt/yt;
        rcx=rc(rho,tau);
    }
    do {
        sqrtx=sqrt(xt);
        sqrty=sqrt(yt);
        sqrtz=sqrt(zt);
        alamb=sqrtx*(sqrty+sqrtz)+sqrty*sqrtz;
        alpha=SQR(pt*(sqrtx+sqrty+sqrtz)+sqrtx*sqrty*sqrtz);
        beta=pt*SQR(pt+alamb);
        sum += fac*rc(alpha,beta);
        fac=0.25*fac;
        xt=0.25*(xt+alamb);
        yt=0.25*(yt+alamb);
        zt=0.25*(zt+alamb);
        pt=0.25*(pt+alamb);
        ave=0.2*(xt+yt+zt+pt+pt);
        delx=(ave-xt)/ave;
        dely=(ave-yt)/ave;
        delz=(ave-zt)/ave;
        delp=(ave-pt)/ave;
    } while (FMAX(FMAX(fabs(delx),fabs(dely)),
        FMAX(fabs(delz),fabs(delp))) > ERRTOL);
    ea=delx*(dely+delz)+dely*delz;
    eb=delx*dely*delz;
    ec=delp*delp;
    ed=ea-3.0*ec;
    ee=eb+2.0*delp*(ea-ec);
    ans=3.0*sum+fac*(1.0+ed*(-C1+C5*ed-C6*ee)+eb*(C7+delp*(-C8+delp*C4))
        +delp*ea*(C2-delp*C3)-C2*delp*ec)/(ave*sqrt(ave));
    if (p <= 0.0) ans=a*(b*ans+3.0*(rcx-rf(xt,yt,zt)));
    return ans;
}
```

```
#include <math.h>
#include "nrutil.h"
#define ERRTOL 0.04
#define TINY 1.69e-38
#define SQRTNY 1.3e-19
#define BIG 3.e37
#define TNBG (TINY*BIG)
#define COMP1 (2.236/SQRTNY)
#define COMP2 (TNBG*TNBG/25.0)
#define THIRD (1.0/3.0)
#define C1 0.3
#define C2 (1.0/7.0)
#define C3 0.375
#define C4 (9.0/22.0)

float rc(float x, float y)
```
Computes Carlson's degenerate elliptic integral, $R_C(x, y)$. x must be nonnegative and y must be nonzero. If $y < 0$, the Cauchy principal value is returned. TINY must be at least 5 times the machine underflow limit, BIG at most one fifth the machine maximum overflow limit.
```
{
    float alamb,ave,s,w,xt,yt;
    if (x < 0.0 || y == 0.0 || (x+fabs(y)) < TINY || (x+fabs(y)) > BIG ||
        (y<-COMP1 && x > 0.0 && x < COMP2))
            nrerror("invalid arguments in rc");
    if (y > 0.0) {
        xt=x;
        yt=y;
        w=1.0;
    } else {
        xt=x-y;
        yt = -y;
        w=sqrt(x)/sqrt(xt);
    }
    do {
        alamb=2.0*sqrt(xt)*sqrt(yt)+yt;
        xt=0.25*(xt+alamb);
        yt=0.25*(yt+alamb);
        ave=THIRD*(xt+yt+yt);
        s=(yt-ave)/ave;
    } while (fabs(s) > ERRTOL);
    return w*(1.0+s*s*(C1+s*(C2+s*(C3+s*C4))))/sqrt(ave);
}
```

At times you may want to express your answer in Legendre's notation. Alternatively, you may be given results in that notation and need to compute their values with the programs given above. It is a simple matter to transform back and forth. The *Legendre elliptic integral of the 1st kind* is defined as

$$F(\phi, k) \equiv \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 \sin^2\theta}} \tag{6.11.17}$$

The *complete elliptic integral of the 1st kind* is given by

$$K(k) \equiv F(\pi/2, k) \tag{6.11.18}$$

In terms of R_F,

$$\begin{aligned} F(\phi, k) &= \sin\phi R_F(\cos^2\phi, 1 - k^2\sin^2\phi, 1) \\ K(k) &= R_F(0, 1 - k^2, 1) \end{aligned} \tag{6.11.19}$$

The *Legendre elliptic integral of the 2nd kind* and the *complete elliptic integral of the 2nd kind* are given by

$$\begin{aligned} E(\phi, k) &\equiv \int_0^\phi \sqrt{1 - k^2 \sin^2 \theta}\, d\theta \\ &= \sin\phi R_F(\cos^2\phi, 1 - k^2 \sin^2\phi, 1) \\ &\qquad - \tfrac{1}{3} k^2 \sin^3\phi R_D(\cos^2\phi, 1 - k^2\sin^2\phi, 1) \\ E(k) &\equiv E(\pi/2, k) = R_F(0, 1-k^2, 1) - \tfrac{1}{3}k^2 R_D(0, 1-k^2, 1) \end{aligned} \tag{6.11.20}$$

Finally, the *Legendre elliptic integral of the 3rd kind* is

$$\begin{aligned} \Pi(\phi, n, k) &\equiv \int_0^\phi \frac{d\theta}{(1 + n\sin^2\theta)\sqrt{1 - k^2\sin^2\theta}} \\ &= \sin\phi R_F(\cos^2\phi, 1 - k^2\sin^2\phi, 1) \\ &\qquad - \tfrac{1}{3} n \sin^3\phi R_J(\cos^2\phi, 1 - k^2\sin^2\phi, 1, 1 + n\sin^2\phi) \end{aligned} \tag{6.11.21}$$

(Note that this sign convention for n is opposite that of Abramowitz and Stegun [12], and that their $\sin\alpha$ is our k.)

```
#include <math.h>
#include "nrutil.h"

float ellf(float phi, float ak)
```
Legendre elliptic integral of the 1st kind $F(\phi, k)$, evaluated using Carlson's function R_F. The argument ranges are $0 \le \phi \le \pi/2$, $0 \le k \sin\phi \le 1$.
```
{
    float rf(float x, float y, float z);
    float s;

    s=sin(phi);
    return s*rf(SQR(cos(phi)),(1.0-s*ak)*(1.0+s*ak),1.0);
}
```

```
#include <math.h>
#include "nrutil.h"

float elle(float phi, float ak)
```
Legendre elliptic integral of the 2nd kind $E(\phi, k)$, evaluated using Carlson's functions R_D and R_F. The argument ranges are $0 \le \phi \le \pi/2$, $0 \le k\sin\phi \le 1$.
```
{
    float rd(float x, float y, float z);
    float rf(float x, float y, float z);
    float cc,q,s;

    s=sin(phi);
    cc=SQR(cos(phi));
    q=(1.0-s*ak)*(1.0+s*ak);
    return s*(rf(cc,q,1.0)-(SQR(s*ak))*rd(cc,q,1.0)/3.0);
}
```

```
#include <math.h>
#include "nrutil.h"

float ellpi(float phi, float en, float ak)
```
Legendre elliptic integral of the 3rd kind $\Pi(\phi, n, k)$, evaluated using Carlson's functions R_J and R_F. (Note that the sign convention on n is opposite that of Abramowitz and Stegun.) The ranges of ϕ and k are $0 \leq \phi \leq \pi/2$, $0 \leq k \sin\phi \leq 1$.
```
{
    float rf(float x, float y, float z);
    float rj(float x, float y, float z, float p);
    float cc,enss,q,s;

    s=sin(phi);
    enss=en*s*s;
    cc=SQR(cos(phi));
    q=(1.0-s*ak)*(1.0+s*ak);
    return s*(rf(cc,q,1.0)-enss*rj(cc,q,1.0,1.0+enss)/3.0);
}
```

Carlson's functions are homogeneous of degree $-\frac{1}{2}$ and $-\frac{3}{2}$, so

$$\begin{aligned} R_F(\lambda x, \lambda y, \lambda z) &= \lambda^{-1/2} R_F(x, y, z) \\ R_J(\lambda x, \lambda y, \lambda z, \lambda p) &= \lambda^{-3/2} R_J(x, y, z, p) \end{aligned} \tag{6.11.22}$$

Thus to express a Carlson function in Legendre's notation, permute the first three arguments into ascending order, use homogeneity to scale the third argument to be 1, and then use equations (6.11.19)–(6.11.21).

Jacobian Elliptic Functions

The Jacobian elliptic function sn is defined as follows: instead of considering the elliptic integral

$$u(y, k) \equiv u = F(\phi, k) \tag{6.11.23}$$

consider the *inverse* function

$$y = \sin\phi = \mathrm{sn}(u, k) \tag{6.11.24}$$

Equivalently,

$$u = \int_0^{\mathrm{sn}} \frac{dy}{\sqrt{(1 - y^2)(1 - k^2 y^2)}} \tag{6.11.25}$$

When $k = 0$, sn is just sin. The functions cn and dn are defined by the relations

$$\mathrm{sn}^2 + \mathrm{cn}^2 = 1, \qquad k^2 \mathrm{sn}^2 + \mathrm{dn}^2 = 1 \tag{6.11.26}$$

The routine given below actually takes $m_c \equiv k_c^2 = 1 - k^2$ as an input parameter. It also computes all three functions sn, cn, and dn since computing all three is no harder than computing any one of them. For a description of the method, see [8].

```
#include <math.h>
#define CA 0.0003                    The accuracy is the square of CA.

void sncndn(float uu, float emmc, float *sn, float *cn, float *dn)
```
Returns the Jacobian elliptic functions $\mathrm{sn}(u, k_c)$, $\mathrm{cn}(u, k_c)$, and $\mathrm{dn}(u, k_c)$. Here $\mathtt{uu} = u$, while $\mathtt{emmc} = k_c^2$.
```
{
    float a,b,c,d,emc,u;
    float em[14],en[14];
    int i,ii,l,bo;

    emc=emmc;
    u=uu;
    if (emc) {
        bo=(emc < 0.0);
        if (bo) {
            d=1.0-emc;
            emc /= -1.0/d;
            u *= (d=sqrt(d));
        }
        a=1.0;
        *dn=1.0;
        for (i=1;i<=13;i++) {
            l=i;
            em[i]=a;
            en[i]=(emc=sqrt(emc));
            c=0.5*(a+emc);
            if (fabs(a-emc) <= CA*a) break;
            emc *= a;
            a=c;
        }
        u *= c;
        *sn=sin(u);
        *cn=cos(u);
        if (*sn) {
            a=(*cn)/(*sn);
            c *= a;
            for (ii=l;ii>=1;ii--) {
                b=em[ii];
                a *= c;
                c *= (*dn);
                *dn=(en[ii]+a)/(b+a);
                a=c/b;
            }
            a=1.0/sqrt(c*c+1.0);
            *sn=(*sn >= 0.0 ? a : -a);
            *cn=c*(*sn);
        }
        if (bo) {
            a=(*dn);
            *dn=(*cn);
            *cn=a;
            *sn /= d;
        }
    } else {
        *cn=1.0/cosh(u);
        *dn=(*cn);
        *sn=tanh(u);
    }
}
```

CITED REFERENCES AND FURTHER READING:

Erdélyi, A., Magnus, W., Oberhettinger, F., and Tricomi, F.G. 1953, *Higher Transcendental Functions*, Vol. II, (New York: McGraw-Hill). [1]

Gradshteyn, I.S., and Ryzhik, I.W. 1980, *Table of Integrals, Series, and Products* (New York: Academic Press). [2]

Carlson, B.C. 1977, *SIAM Journal on Mathematical Analysis*, vol. 8, pp. 231–242. [3]

Carlson, B.C. 1987, *Mathematics of Computation*, vol. 49, pp. 595–606 [4]; 1988, *op. cit.*, vol. 51, pp. 267–280 [5]; 1989, *op. cit.*, vol. 53, pp. 327–333 [6]; 1991, *op. cit.*, vol. 56, pp. 267–280. [7]

Bulirsch, R. 1965, *Numerische Mathematik*, vol. 7, pp. 78–90; 1965, *op. cit.*, vol. 7, pp. 353–354; 1969, *op. cit.*, vol. 13, pp. 305–315. [8]

Carlson, B.C. 1979, *Numerische Mathematik*, vol. 33, pp. 1–16. [9]

Carlson, B.C., and Notis, E.M. 1981, *ACM Transactions on Mathematical Software*, vol. 7, pp. 398–403. [10]

Carlson, B.C. 1978, *SIAM Journal on Mathematical Analysis*, vol. 9, p. 524–528. [11]

Abramowitz, M., and Stegun, I.A. 1964, *Handbook of Mathematical Functions*, Applied Mathematics Series, Volume 55 (Washington: National Bureau of Standards; reprinted 1968 by Dover Publications, New York), Chapter 17. [12]

Mathews, J., and Walker, R.L. 1970, *Mathematical Methods of Physics*, 2nd ed. (Reading, MA: W.A. Benjamin/Addison-Wesley), pp. 78–79.

6.12 Hypergeometric Functions

As was discussed in §5.14, a fast, general routine for the the complex hypergeometric function ${}_2F_1(a,b,c;z)$, is difficult or impossible. The function is defined as the analytic continuation of the hypergeometric series,

$$\begin{aligned} {}_2F_1(a,b,c;z) = 1 &+ \frac{ab}{c}\frac{z}{1!} + \frac{a(a+1)b(b+1)}{c(c+1)}\frac{z^2}{2!} + \cdots \\ &+ \frac{a(a+1)\dots(a+j-1)b(b+1)\dots(b+j-1)}{c(c+1)\dots(c+j-1)}\frac{z^j}{j!} + \cdots \end{aligned} \tag{6.12.1}$$

This series converges only within the unit circle $|z| < 1$ (see [1]), but one's interest in the function is not confined to this region.

Section 5.14 discussed the method of evaluating this function by direct path integration in the complex plane. We here merely list the routines that result.

Implementation of the function `hypgeo` is straightforward, and is described by comments in the program. The machinery associated with Chapter 16's routine for integrating differential equations, `odeint`, is only minimally intrusive, and need not even be completely understood: use of `odeint` requires one zeroed global variable, one function call, and a prescribed format for the derivative routine `hypdrv`.

The function `hypgeo` will fail, of course, for values of z too close to the singularity at 1. (If you need to approach this singularity, or the one at ∞, use the "linear transformation formulas" in §15.3 of [1].) Away from $z = 1$, and for moderate values of a, b, c, it is often remarkable how few steps are required to integrate the equations. A half-dozen is typical.