6.12 Hypergeometric Functions

As was discussed in §5.14, a fast, general routine for 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,

\[
_2F_1(a, b; c; z) = 1 + \frac{ab}{c} \frac{z}{1!} + \frac{a(a+1)b(b+1)}{c(c+1)} \frac{z^2}{2!} + \cdots + \frac{a(a+1)\cdots(a+j-1)b(b+1)\cdots(b+j-1)}{c(c+1)\cdots(c+j-1)} \frac{z^j}{j!} + \cdots
\]

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

The function \( \text{hypgeo} \) 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 \( \infty \), 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.
#include <math.h>
#include "complex.h"
#include "nrutil.h"
#define EPS 1.0e-6

fcomplex aa, bb, cc, z0, dz;

fcomplex hypgeo(fcomplex a, fcomplex b, fcomplex c, fcomplex z)
Complex hypergeometric function \( \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b-c+1)} F_1 \) for complex \( a, b, c \), and \( z \), by direct integration of the hypergeometric equation in the complex plane. The branch cut is taken to lie along the real axis, \( \text{Re} z > 1 \).
{
    void bsstep(float y[], float dydx[], int nv, float *xx, float htry,
                void (*derivs)(float, float [], float [], float *hnext),
                void (*hypdrv)(float s, float yy[], float dyyds[],
                               void (*hypser)(fcomplex a, fcomplex b, fcomplex c, fcomplex z,
                                                  fcomplex *series, fcomplex *deriv);
                void odeint(float ystart[], int nvar, float x1, float x2,
                             float eps, float h1, float hmin, int *nok, int *nbad,
                             void (*derivs)(float, float [], float [],
                                            void (*rkqs)(float [], float [], int, float *, float, float,
                                                         float [], float *, float *, void (*)(float, float [], float [])),
                             int nbad,nok;
    fcomplex ans,y[3];
    float *yy;
    kmax=0;
    if (z.r*z.r+z.i*z.i <= 0.25) {  // Use series...
        hypser(a,b,c,z,&ans,&y[2];
        return ans;
    }
    else if (z.r < 0.0) z0=Complex(-0.5,0.0);  // ...or pick a starting point for the path
    else if (z.r <= 1.0) z0=Complex(0.5,0.0);  // by integration.
    else z0=Complex(0.0,z.i >= 0.0 ? 0.5 : -0.5);
    aa=a;
    bb=b;
    cc=c;
    dz=Csub(z,z0);
    hypser(aa,bb,cc,z0,&y[1],&y[2]);  // Get starting function and derivative.
    yy=vector(1,4);  // y=vector(1,4);
    yy[1]=y[1].r;
    yy[2]=y[1].i;
    odeint(yy,4,0.0,1.0,EPS,0.1,0.0001,&nok,&nbad,hypdrv,bsstep);
    y[1]=Complex(yy[1],yy[2]);
    free_vector(yy,1,4);
    return y[1];
}
#include "complex.h"
#define ONE Complex(1.0,0.0)

void hypser(fcomplex a, fcomplex b, fcomplex c, fcomplex z, fcomplex *series, 
    fcomplex *deriv)
Returns the hypergeometric series \( \hyper{2}{1} \) and its derivative, iterating to machine accuracy. For 
\( |z| \leq 1/2 \) convergence is quite rapid.
{
    void nrerror(char error_text[]);
    int n;
    fcomplex aa, bb, cc, fac, temp;
    deriv->r=0.0;
    deriv->i=0.0;
    fac=Complex(1.0,0.0);
    temp=fac;
    aa=a;
    bb=b;
    cc=c;
    for (n=1;n<=1000;n++) {
        fac=Cmul(fac,Cdiv(Cmul(aa,bb),cc));
        deriv->r+=fac.r;
        deriv->i+=fac.i;
        fac=Cmul(fac,RCmul(1.0/n,z));
        *series=Cadd(temp,fac);
        if (series->r == temp.r && series->i == temp.i) return;
        temp= *series;
        aa=Cadd(aa,ONE);
        bb=Cadd(bb,ONE);
        cc=Cadd(cc,ONE);
    }
    nrerror("convergence failure in hypser");
}

#include "complex.h"
#define ONE Complex(1.0,0.0)

extern fcomplex aa, bb, cc, z0, dz;  Defined in hypgeo.

void hypdrv(float s, float yy[], float dyyds[])
Computes derivatives for the hypergeometric equation, see text equation (5.14.4).
{
    fcomplex z, y[3], dyds[3];
    y[1]=Complex(yy[1],yy[2]);
    y[2]=Complex(yy[3],yy[4]);
    z=Cadd(z0,RCmul(s,dz));
    dyds[1]=Cmul(y[2],dz);
    dyds[2]=Cmul(Csub(Cmul(Cmul(aa,bb),y[1]),Cmul(Csub(cc,
        Cmul(Cadd(Cadd(aa,bb),ONE),z),y[2])), 
        Cdiv(dz,Cmul(z,Csub(ONE,z)))));
    dyds[1]=dyds[1].r;
}

CITED REFERENCES AND FURTHER READING: