Hypergeometric Functions by Direct Path Integration

William H. Press, and Saul A. Teukolsky

Citation: Computers in Physics **4**, 320 (1990); doi: 10.1063/1.4822917
View online: https://doi.org/10.1063/1.4822917
View Table of Contents: http://aip.scitation.org/toc/cip/4/3
Published by the American Institute of Physics

Hypergeometric Functions By Direct Path Integration

William H. Press and Saul A. Teukolsky

The proverb, "the longest way round is the shortest way home," dating from the 17th century, applies as well to present-day computer programming. The technique of choice in many real-life computations is not necessarily the most efficient or elegant one, but may instead be the one that is quick to program and easy to check.

An example is the calculation of a special function, perhaps a complex valued function of a complex variable, that has many different parameters, or asymptotic regimes, or both. Use of the usual tricks (series, continued fractions, rational function approximations, recurrence relations, and so forth) may result in a patchwork program with tests and branches to different formulas. While such a program may be highly efficient in execution, it is frequently not the "shortest way home" if you need to write and debug it yourself just to get a few (or a few thousand) function values.

A different technique of considerable generality is direct integration of a function's defining differential equation—an *ab initio* integration for each desired function value— along a path in the complex plane if necessary. While this may at first seem like swatting a fly with a golden brick, it turns out that when you already have the brick, and the fly is asleep right under it, all you have to do is let it fall!

As a specific example, let us consider the complex hypergeometric function ${}_2F_1(a,b,c;z)$, which is defined as the analytic continuation of the so-called hypergeometric series,

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

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

The hypergeometric function ${}_2F_1$ is a solution (in fact, *the* solution that is regular at the origin) of the hypergeometric differential equation, which we can write as

$$z(1-z)F'' = abF - [c - (a+b+1)z]F'. \qquad (2)$$

William H. Press is a professor of astronomy and physics at Harvard University. Saul A. Teukolsky is a professor of physics and astronomy at Cornell University.

Here, prime denotes d/dz. One can see that the equation has regular singular points at $z = 0,1$, and ∞. Since the desired solution is regular at $z = 0$, the values 1 and ∞ will in general be branch points. If we want ${}_2F_1$ to be a single-valued function, we must have a branch cut connecting these two points. A conventional position for this cut is along the positive real axis from 1 to ∞, though we may wish to keep open the possibility of altering this choice for some applications.

Let us look now to our golden brick, namely, the elaborate machinery that we have available for the integration of a set of ordinary differential equations. Most subroutine libraries contain a high-level, "black-box" routine that integrates such a set from initial conditions at one value of a (real) independent variable to final conditions at some other value of the independent variable, while automatically adjusting its internal step size to maintain some specified accuracy. Not unexpectedly, we are partial to the routine in the *Numerical Recipes*[2-4] books, which is called odeint and, in one incarnation, calculates its individual steps with a sophisticated Bulirsch–Stoer technique.

Suppose that we know values for F and its derivative F' at some value z_0, and that we want to find F at some other point z_1 in the complex plane. The straight-line path connecting these two points is parametrized by

$$z(s) = z_0 + s(z_1 - z_0), \qquad (3)$$

with s a real parameter. The differential equation (2) can now be written as a set of two first-order equations,

$$
\begin{aligned}
\frac{dF}{ds} &= (z_1 - z_0)F', \\
\frac{dF'}{ds} &= (z_1 - z_0)\left(\frac{abF - [c - (a+b+1)z]F'}{z(1-z)}\right),
\end{aligned} \qquad (4)
$$

to be integrated $s = 0$ to $s = 1$. Here, F and F' are to be viewed as two independent complex variables; the fact that prime means d/dz can be ignored. Moreover, the real and imaginary parts of Eq. (4) define a set of four *real* differential equations, with independent variable s. The compex arithmetic on the right-hand side can be viewed as mere shorthand for how the four components are to be coupled. It is precisely this point of view that gets passed to the routine odeint, since it knows nothing of either complex functions or complex independent variables.

It remains only to decide where to start, and what path to take in the complex plane, to get to an arbitrary point z. This is where consideration of the function's singularities, and the adopted branch cut, enter. Figure 1

shows the strategy that we adopt. For $|z| \leq 1/2$, the series in Eq. (1) will in general converge rapidly, and it makes sense to use it directly. Otherwise, we integrate along a straight-line path from one of the starting points ($\pm 1/2$, 0) or $(0, \pm 1/2)$. The former choices are natural for $0 < \mathrm{Re}(z) < 1$ and $\mathrm{Re}(z) < 0$, respectively. The latter choices are used for $\mathrm{Re}(z) > 1$, above and below the branch cut; the purpose of starting away from the real axis in these cases is to avoid passing too close to the singularity at $z = 1$ (see Fig. 1). The location of the branch cut is *defined* by the fact that our adopted strategy never integrates across the real axis for $\mathrm{Re}(z) > 1$.

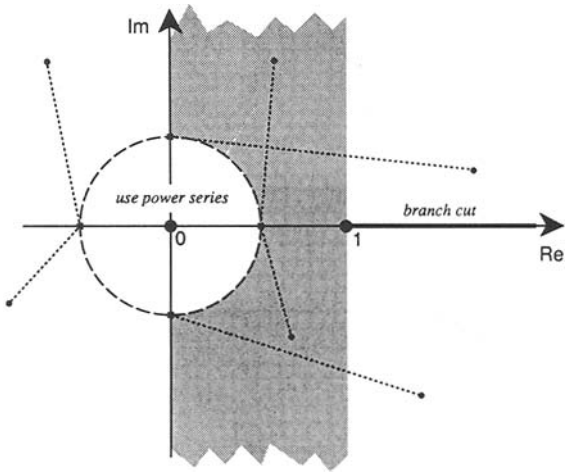

FIG. 1. Complex plane showing the singular points of the hypergeometric function, its branch cut, and some integration paths from the circle $|z| = \frac{1}{2}$ (where the power series converges rapidly) to other points in the plane.

Implementation of the subroutine hypgeo (Box 1) is now straightforward, and is described by comments in the program. The machinery associated with odeint is only minimally intrusive, and need not even be completely understood: a common block with one zeroed variable, one subroutine call, and a prescribed format for the derivative routine hypdrv.

The subroutine 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 Sec. 15.3 of Ref. 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.

A number of variants on the procedure described thus far are possible, and easy to program. If successively called values of z are close together (with identical values of a, b, and c), then you can save the state vector y and the corresponding value of z on each call, and use them as starting values for the next call. The incremental integration may then take only one or two steps. Avoid integrating across the branch cut unintentionally: The function value will be "correct," but not the one you want.

Alternatively, you may wish to integrate to some position by a dog-leg path that *does* cross the real axis

```
COMPLEX FUNCTION hypgeo(a,b,c,z)
COMPLEX a,b,c,z, aa,bb,cc,z0,dz
EXTERNAL bsstep,hypdrv
REAL EPS
PARAMETER (EPS=1.e-6)          Accuracy parameter.
COMMON /hypg/ aa,bb,cc,z0,dz
USES bsstep,hypdrv,hypser,odeint
  Complex hypergeometric function 2F1 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, Re z > 1.
COMPLEX y(2)
INTEGER nbad,nok,kmax
COMMON /path/ kmax             Used by odeint.
kmax=0
if (real(z)**2+aimag(z)**2.le.0.25) then      Use series...
    call hypser(a,b,c,z,hypgeo,y(2))
    return
else if (real(z).lt.0.) then          ...or pick a starting point for the path integration.
    z0=cmplx(-0.5,0.)
else if (real(z).le.1.0) then
    z0=cmplx(0.5,0.)
else
    z0=cmplx(0.,sign(0.5,aimag(z)))
endif
aa=a                           Load the common block, used to pass parameters "over the
bb=b                           head" of odeint to hypdrv.
cc=c
dz=z-z0
call hypser(aa,bb,cc,z0,y(1),y(2))        Get starting function and derivative.
call odeint(y,4,0.,1.,EPS,.1,.0001,nok,nbad,hypdrv,bsstep)
  The arguments to odeint are the vector of independent variables, its length, the starting and
  ending values of the dependent variable, the accuracy parameter, an initial guess for step size,
  a minimum step size, the (returned) number of good and bad steps taken, and the names of
  the derivative routine and the (here Bulirsch-Stoer) stepping routine.
hypgeo=y(1)
return
END

SUBROUTINE hypdrv(s,y,dyds)
REAL s
COMPLEX y(2),dyds(2), aa,bb,cc,z0,dz
COMMON /hypg/ aa,bb,cc,z0,dz
  Derivative subroutine for the hypergeometric equation, see text equation (4).
COMPLEX z
z=z0+s*dz
dyds(1)=y(2)*dz
dyds(2)=(aa*bb*y(1)-(cc-(aa+bb+1.)*z)*y(2))*dz/(z*(1.-z))
return
END

SUBROUTINE hypser(a,b,c,z,series,deriv)
COMPLEX a,b,c,z,series,deriv
  Returns the hypergeometric series 2F1 and its derivative, iterating to machine accuracy. For
  cabs(z)≤ 1/2 convergence is quite rapid.
INTEGER n
COMPLEX aa,bb,cc,fac,temp
deriv=cmplx(0.,0.)
fac=cmplx(1.,0.)
temp=fac
aa=a
bb=b
cc=c
do 11 n=1,1000
    fac=fac*aa*bb/cc
    deriv=deriv+fac
    fac=fac*z/n
    series=temp+fac
    if (series.eq.temp) return
    temp=series
    aa=aa+1.
    bb=bb+1.
    cc=cc+1.
enddo 11
pause 'convergence failure in hypser'
END
```

$\mathrm{Re}\, z > 1$, as a means of *moving* the branch cut. For example, in some cases you might want to integrate from (0, 1/2) to (3/2, 1/2), and go from there to any point with $\mathrm{Re}\, z > 1$. (If you are, for example, finding roots of a function by an iterative method, you do not want the integration for nearby values to take different paths around a branch point. If it does, your root finder will see discontinuous function values, and will likely not converge correctly!)

The general technique of evaluating a function by integrating its differential equation in the complex plane can also be applied to other special functions. For example, the complex Bessel function, Airy function, Coulomb wavefunction, and Weber function are all special cases of the *confluent hypergeometric function*, with a differential equation similar to the one used above (see, e.g., Ref. 1, Sec. 13.6, for a table of special cases). The confluent hypergeometric function has no singularities at finite z: That makes it easy to integrate. However, its essential singularity at infinity means that it can have, along some paths and for some parameters, highly oscillatory or exponentially decreasing behavior: That makes it hard to integrate. Some case-by-case judgment (or experimentation) is therefore required. ■

References

1. M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions* (National Bureau of Standards, Washington, DC, 1964).
2. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing* (Cambridge U.P., New York, 1986).
3. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing* (Cambridge U.P., New York, 1988).
4. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in Pascal: The Art of Scientific Computing* (Cambridge U.P., New York, 1989).

In our next column: Orthogonal polynomials and Gaussian quadratures with nonclassical weights.