Figure 5.13.1 shows the discrepancies for the first five iterations of `ratlsq` when it is applied to find the $m = k = 4$ rational fit to the function $f(x) = \cos x/(1 + e^x)$ in the interval $(0, \pi)$. One sees that after the first iteration, the results are virtually as good as the minimax solution. The iterations do not converge in the order that the figure suggests: In fact, it is the second iteration that is best (has smallest maximum deviation). The routine `ratlsq` accordingly returns the best of its iterations, not necessarily the last one; there is no advantage in doing more than five iterations.

CITED REFERENCES AND FURTHER READING:

Ralston, A. and Wilf, H.S. 1960, *Mathematical Methods for Digital Computers* (New York: Wiley), Chapter 13. [1]

5.14 Evaluation of Functions by Path Integration

In computer programming, the technique of choice is not necessarily the most efficient, or elegant, or fastest executing one. Instead, it may be the one that is quick to implement, general, and easy to check.

One sometimes needs only a few, or a few thousand, evaluations 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 often not the shortest way to the answer from a standing start.

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,

$$\begin{aligned}{}_2F_1(a, b, c; z) = 1 &+ \frac{ab}{c}\frac{z}{1!} + \frac{a(a+1)b(b+1)}{c(c+1)}\frac{z^2}{2!} + \cdots \\ &+ \frac{a(a+1)\ldots(a+j-1)b(b+1)\ldots(b+j-1)}{c(c+1)\ldots(c+j-1)}\frac{z^j}{j!} + \cdots\end{aligned} \tag{5.14.1}$$

The series converges only within the unit circle $|z| < 1$ (see [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

$$z(1-z)F'' = abF - [c - (a+b+1)z]F' \tag{5.14.2}$$

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.

Our golden brick consists of a collection of routines for the integration of sets of ordinary differential equations, which we will develop in detail later, in Chapter 16. For now, we need only 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 stepsize to maintain some specified accuracy. That routine is called `odeint` and, in one particular invocation, 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

$$z(s) = z_0 + s(z_1 - z_0) \tag{5.14.3}$$

with s a real parameter. The differential equation (5.14.2) can now be written as a set of two first-order equations,

$$\begin{aligned}
\frac{dF}{ds} &= (z_1 - z_0)F' \\
\frac{dF'}{ds} &= (z_1 - z_0)\left(\frac{abF - [c - (a+b+1)z]F'}{z(1-z)}\right)
\end{aligned} \tag{5.14.4}$$

to be integrated from $s = 0$ 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; it will emerge as a consequence of the first equation in (5.14.4). Moreover, the real and imaginary parts of equation (5.14.4) define a set of four *real* differential equations, with independent variable s. The complex 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 5.14.1 shows the strategy that we adopt. For $|z| \leq 1/2$, the series in equation (5.14.1) 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 $0 < \text{Re}(z) < 1$ and $\text{Re}(z) < 0$, respectively. The latter choices are used for $\text{Re}(z) > 1$, 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 Figure 5.14.1). The location of the branch cut is *defined* by the fact that our adopted strategy never integrates across the real axis for Re $(z) > 1$.

An implementation of this algorithm is given in §6.12 as the routine `hypgeo`.

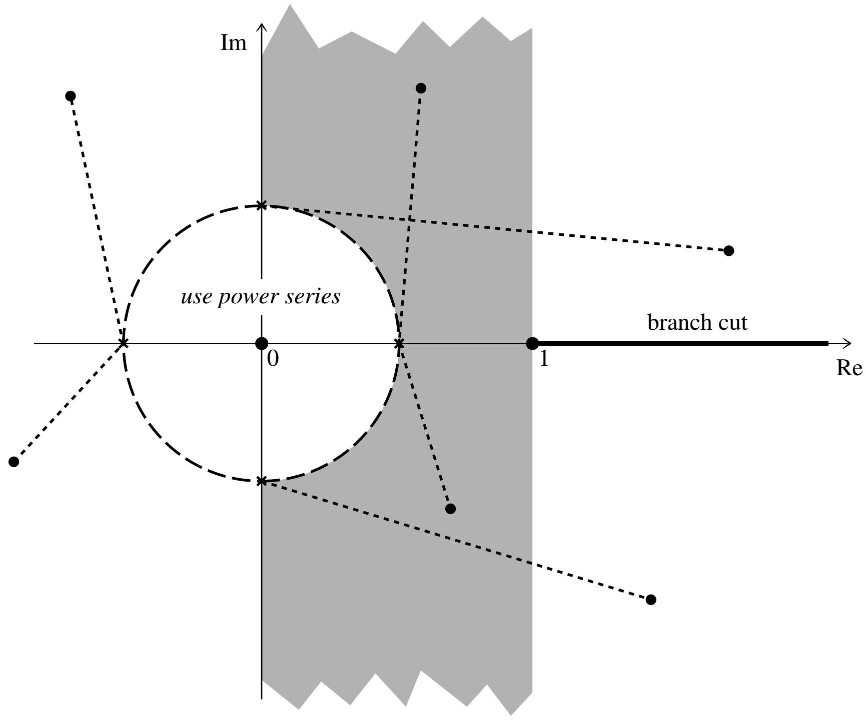

Figure 5.14.1. Complex plane showing the singular points of the hypergeometric function, its branch cut, and some integration paths from the circle $|z| = 1/2$ (where the power series converges rapidly) to other points in the plane.

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 (F, F') and the corresponding value of z on each call, and use these 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 z by a dog-leg path that *does* cross the real axis 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 Re $z > 1$ — with either sign of Im z. (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!)

In any case, be aware that a loss of numerical accuracy can result if you integrate through a region of large function value on your way to a final answer where the function value is small. (For the hypergeometric function, a particular case of this is when a and b are both large and positive, with c and $x \gtrsim 1$.) In such cases, you'll need to find a better dog-leg path.

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 wave function, 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., [1] §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.

CITED REFERENCES AND FURTHER READING:

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). [1]