into equation (5.1.11), and then setting $z = 1$.

Sometimes you will want to compute a function from a series representation even when the computation is *not* efficient. For example, you may be using the values obtained to fit the function to an approximating form that you will use subsequently (cf. §5.8). If you are summing very large numbers of slowly convergent terms, pay attention to roundoff errors! In floating-point representation it is more accurate to sum a list of numbers in the order starting with the smallest one, rather than starting with the largest one. It is even better to group terms pairwise, then in pairs of pairs, etc., so that all additions involve operands of comparable magnitude.

CITED REFERENCES AND FURTHER READING:

Goodwin, E.T. (ed.) 1961, *Modern Computing Methods*, 2nd ed. (New York: Philosophical Library), Chapter 13 [van Wijngaarden's transformations]. [1]

Dahlquist, G., and Bjorck, A. 1974, *Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall), Chapter 3.

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), §3.6.

Mathews, J., and Walker, R.L. 1970, *Mathematical Methods of Physics*, 2nd ed. (Reading, MA: W.A. Benjamin/Addison-Wesley), §2.3. [2]

5.2 Evaluation of Continued Fractions

Continued fractions are often powerful ways of evaluating functions that occur in scientific applications. A continued fraction looks like this:

$$f(x) = b_0 + \cfrac{a_1}{b_1 + \cfrac{a_2}{b_2 + \cfrac{a_3}{b_3 + \cfrac{a_4}{b_4 + \cfrac{a_5}{b_5 + \cdots}}}}} \tag{5.2.1}$$

Printers prefer to write this as

$$f(x) = b_0 + \frac{a_1}{b_1 +}\,\frac{a_2}{b_2 +}\,\frac{a_3}{b_3 +}\,\frac{a_4}{b_4 +}\,\frac{a_5}{b_5 +}\cdots \tag{5.2.2}$$

In either (5.2.1) or (5.2.2), the a's and b's can themselves be functions of x, usually linear or quadratic monomials at worst (i.e., constants times x or times x^2). For example, the continued fraction representation of the tangent function is

$$\tan x = \frac{x}{1 -}\,\frac{x^2}{3 -}\,\frac{x^2}{5 -}\,\frac{x^2}{7 -}\cdots \tag{5.2.3}$$

Continued fractions frequently converge much more rapidly than power series expansions, and in a much larger domain in the complex plane (not necessarily including the domain of convergence of the series, however). Sometimes the continued fraction converges best where the series does worst, although this is not a general rule. Blanch [1] gives a good review of the most useful convergence tests for continued fractions.

There are standard techniques, including the important *quotient-difference algorithm*, for going back and forth between continued fraction approximations, power series approximations, and rational function approximations. Consult Acton [2] for an introduction to this subject, and Fike [3] for further details and references.

How do you tell how far to go when evaluating a continued fraction? Unlike a series, you can't just evaluate equation (5.2.1) from left to right, stopping when the change is small. Written in the form of (5.2.1), the only way to evaluate the continued fraction is from right to left, first (blindly!) guessing how far out to start. This is not the right way.

The right way is to use a result that relates continued fractions to rational approximations, and that gives a means of evaluating (5.2.1) or (5.2.2) from left to right. Let f_n denote the result of evaluating (5.2.2) with coefficients through a_n and b_n. Then

$$f_n = \frac{A_n}{B_n} \tag{5.2.4}$$

where A_n and B_n are given by the following recurrence:

$$\begin{gathered} A_{-1} \equiv 1 \qquad B_{-1} \equiv 0 \\ A_0 \equiv b_0 \qquad B_0 \equiv 1 \\ A_j = b_j A_{j-1} + a_j A_{j-2} \qquad B_j = b_j B_{j-1} + a_j B_{j-2} \qquad j = 1, 2, \ldots, n \end{gathered} \tag{5.2.5}$$

This method was invented by J. Wallis in 1655 (!), and is discussed in his *Arithmetica Infinitorum* [4]. You can easily prove it by induction.

In practice, this algorithm has some unattractive features: The recurrence (5.2.5) frequently generates very large or very small values for the partial numerators and denominators A_j and B_j. There is thus the danger of overflow or underflow of the floating-point representation. However, the recurrence (5.2.5) is linear in the A's and B's. At any point you can rescale the currently saved two levels of the recurrence, e.g., divide A_j, B_j, A_{j-1}, and B_{j-1} all by B_j. This incidentally makes $A_j = f_j$ and is convenient for testing whether you have gone far enough: See if f_j and f_{j-1} from the last iteration are as close as you would like them to be. (If B_j happens to be zero, which can happen, just skip the renormalization for this cycle. A fancier level of optimization is to renormalize only when an overflow is imminent, saving the unnecessary divides. All this complicates the program logic.)

Two newer algorithms have been proposed for evaluating continued fractions. *Steed's method* does not use A_j and B_j explicitly, but only the *ratio* $D_j = B_{j-1}/B_j$. One calculates D_j and $\Delta f_j = f_j - f_{j-1}$ recursively using

$$D_j = 1/(b_j + a_j D_{j-1}) \tag{5.2.6}$$

$$\Delta f_j = (b_j D_j - 1)\Delta f_{j-1} \tag{5.2.7}$$

Steed's method (see, e.g., [5]) avoids the need for rescaling of intermediate results. However, for certain continued fractions you can occasionally run into a situation where the denominator in (5.2.6) approaches zero, so that D_j and Δf_j are very large. The next Δf_{j+1} will typically cancel this large change, but with loss of accuracy in the numerical running sum of the f_j's. It is awkward to program around this, so Steed's method can be recommended only for cases where you know in advance that no denominator can vanish. We will use it for a special purpose in the routine `bessik` (§6.7).

The best general method for evaluating continued fractions seems to be the *modified Lentz's method* [6]. The need for rescaling intermediate results is avoided by using *both* the ratios

$$C_j = A_j/A_{j-1}, \qquad D_j = B_{j-1}/B_j \tag{5.2.8}$$

and calculating f_j by

$$f_j = f_{j-1}C_jD_j \tag{5.2.9}$$

From equation (5.2.5), one easily shows that the ratios satisfy the recurrence relations

$$D_j = 1/(b_j + a_jD_{j-1}), \qquad C_j = b_j + a_j/C_{j-1} \tag{5.2.10}$$

In this algorithm there is the danger that the denominator in the expression for D_j, or the quantity C_j itself, might approach zero. Either of these conditions invalidates (5.2.10). However, Thompson and Barnett [5] show how to modify Lentz's algorithm to fix this: Just shift the offending term by a small amount, e.g., 10^{-30}. If you work through a cycle of the algorithm with this prescription, you will see that f_{j+1} is accurately calculated.

In detail, the modified Lentz's algorithm is this:

- Set $f_0 = b_0$; if $b_0 = 0$ set $f_0 = tiny$.
- Set $C_0 = f_0$.
- Set $D_0 = 0$.
- For $j = 1, 2, \ldots$
  - Set $D_j = b_j + a_jD_{j-1}$.
  - If $D_j = 0$, set $D_j = tiny$.
  - Set $C_j = b_j + a_j/C_{j-1}$.
  - If $C_j = 0$ set $C_j = tiny$.
  - Set $D_j = 1/D_j$.
  - Set $\Delta_j = C_jD_j$.
  - Set $f_j = f_{j-1}\Delta_j$.
  - If $|\Delta_j - 1| < eps$ then exit.

Here eps is your floating-point precision, say 10^{-7} or 10^{-15}. The parameter $tiny$ should be less than typical values of $eps|b_j|$, say 10^{-30}.

The above algorithm assumes that you can terminate the evaluation of the continued fraction when $|f_j - f_{j-1}|$ is sufficiently small. This is usually the case, but by no means guaranteed. Jones [7] gives a list of theorems that can be used to justify this termination criterion for various kinds of continued fractions.

There is at present no rigorous analysis of error propagation in Lentz's algorithm. However, empirical tests suggest that it is at least as good as other methods.

Manipulating Continued Fractions

Several important properties of continued fractions can be used to rewrite them in forms that can speed up numerical computation. An *equivalence transformation*

$$a_n \to \lambda a_n, \quad b_n \to \lambda b_n, \quad a_{n+1} \to \lambda a_{n+1} \tag{5.2.11}$$

leaves the value of a continued fraction unchanged. By a suitable choice of the scale factor λ you can often simplify the form of the a's and the b's. Of course, you can carry out successive equivalence transformations, possibly with different λ's, on successive terms of the continued fraction.

The *even* and *odd* parts of a continued fraction are continued fractions whose successive convergents are f_{2n} and f_{2n+1}, respectively. Their main use is that they converge twice as fast as the original continued fraction, and so if their terms are not much more complicated than the terms in the original there can be a big savings in computation. The formula for the even part of (5.2.2) is

$$f_{\text{even}} = d_0 + \frac{c_1}{d_1+}\,\frac{c_2}{d_2+}\,\cdots \tag{5.2.12}$$

where in terms of intermediate variables

$$\begin{aligned} \alpha_1 &= \frac{a_1}{b_1} \\ \alpha_n &= \frac{a_n}{b_n b_{n-1}}, \qquad n \geq 2 \end{aligned} \tag{5.2.13}$$

we have

$$\begin{aligned} d_0 = b_0, \quad c_1 = \alpha_1, \quad d_1 = 1 + \alpha_2 \\ c_n = -\alpha_{2n-1}\alpha_{2n-2}, \quad d_n = 1 + \alpha_{2n-1} + \alpha_{2n}, \qquad n \geq 2 \end{aligned} \tag{5.2.14}$$

You can find the similar formula for the odd part in the review by Blanch [1]. Often a combination of the transformations (5.2.14) and (5.2.11) is used to get the best form for numerical work.

We will make frequent use of continued fractions in the next chapter.

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), §3.10.

Blanch, G. 1964, *SIAM Review*, vol. 6, pp. 383–421. [1]

Acton, F.S. 1970, *Numerical Methods That Work*; 1990, corrected edition (Washington: Mathematical Association of America), Chapter 11. [2]

Cuyt, A., and Wuytack, L. 1987, *Nonlinear Methods in Numerical Analysis* (Amsterdam: North-Holland), Chapter 1.

Fike, C.T. 1968, *Computer Evaluation of Mathematical Functions* (Englewood Cliffs, NJ: Prentice-Hall), §§8.2, 10.4, and 10.5. [3]

Wallis, J. 1695, in *Opera Mathematica*, vol. 1, p. 355, Oxoniae e Theatro Shedoniano. Reprinted by Georg Olms Verlag, Hildeshein, New York (1972). [4]

Thompson, I.J., and Barnett, A.R. 1986, *Journal of Computational Physics*, vol. 64, pp. 490–509. [5]

Lentz, W.J. 1976, *Applied Optics*, vol. 15, pp. 668–671. [6]

Jones, W.B. 1973, in *Padé Approximants and Their Applications*, P.R. Graves-Morris, ed. (London: Academic Press), p. 125. [7]

5.3 Polynomials and Rational Functions

A polynomial of degree N is represented numerically as a stored array of coefficients, `c[j]` with `j`$= 0, \ldots, N$. We will always take `c[0]` to be the constant term in the polynomial, `c[`N`]` the coefficient of x^N; but of course other conventions are possible. There are two kinds of manipulations that you can do with a polynomial: *numerical* manipulations (such as evaluation), where you are given the numerical value of its argument, or *algebraic* manipulations, where you want to transform the coefficient array in some way without choosing any particular argument. Let's start with the numerical.

We assume that you know enough *never* to evaluate a polynomial this way:

```
p=c[0]+c[1]*x+c[2]*x*x+c[3]*x*x*x+c[4]*x*x*x*x;
```

or (even worse!),

```
p=c[0]+c[1]*x+c[2]*pow(x,2.0)+c[3]*pow(x,3.0)+c[4]*pow(x,4.0);
```

Come the (computer) revolution, all persons found guilty of such criminal behavior will be summarily executed, and their programs won't be! It is a matter of taste, however, whether to write

```
p=c[0]+x*(c[1]+x*(c[2]+x*(c[3]+x*c[4])));
```

or

```
p=(((c[4]*x+c[3])*x+c[2])*x+c[1])*x+c[0];
```

If the number of coefficients `c[0..n]` is large, one writes

```
p=c[n];
for(j=n-1;j>=0;j--) p=p*x+c[j];
```

or

```
p=c[j=n];
while (j>0) p=p*x+c[--j];
```

Another useful trick is for evaluating a polynomial $P(x)$ and its derivative $dP(x)/dx$ simultaneously:

```
p=c[n];
dp=0.0;
for(j=n-1;j>=0;j--) {dp=dp*x+p; p=p*x+c[j];}
```