3.2 Rational Function Interpolation and Extrapolation

Some functions are not well approximated by polynomials, but *are* well approximated by rational functions, that is quotients of polynomials. We denote by $R_{i(i+1)\dots(i+m)}$ a rational function passing through the $m+1$ points $(x_i, y_i)\dots(x_{i+m}, y_{i+m})$. More explicitly, suppose

$$R_{i(i+1)\dots(i+m)} = \frac{P_\mu(x)}{Q_\nu(x)} = \frac{p_0 + p_1 x + \cdots + p_\mu x^\mu}{q_0 + q_1 x + \cdots + q_\nu x^\nu} \tag{3.2.1}$$

Since there are $\mu + \nu + 1$ unknown p's and q's (q_0 being arbitrary), we must have

$$m + 1 = \mu + \nu + 1 \tag{3.2.2}$$

In specifying a rational function interpolating function, you must give the desired order of both the numerator and the denominator.

Rational functions are sometimes superior to polynomials, roughly speaking, because of their ability to model functions with poles, that is, zeros of the denominator of equation (3.2.1). These poles might occur for real values of x, if the function to be interpolated itself has poles. More often, the function $f(x)$ is finite for all finite *real* x, but has an analytic continuation with poles in the complex x-plane. Such poles can themselves ruin a polynomial approximation, even one restricted to real values of x, just as they can ruin the convergence of an infinite power series in x. If you draw a circle in the complex plane around your m tabulated points, then you should not expect polynomial interpolation to be good unless the nearest pole is rather far outside the circle. A rational function approximation, by contrast, will stay "good" as long as it has enough powers of x in its denominator to account for (cancel) any nearby poles.

For the interpolation problem, a rational function is constructed so as to go through a chosen set of tabulated functional values. However, we should also mention in passing that rational function approximations can be used in analytic work. One sometimes constructs a rational function approximation by the criterion that the rational function of equation (3.2.1) itself have a power series expansion that agrees with the first $m+1$ terms of the power series expansion of the desired function $f(x)$. This is called *Padé approximation*, and is discussed in §5.12.

Bulirsch and Stoer found an algorithm of the Neville type which performs rational function extrapolation on tabulated data. A tableau like that of equation (3.1.2) is constructed column by column, leading to a result and an error estimate. The Bulirsch-Stoer algorithm produces the so-called *diagonal* rational function, with the degrees of numerator and denominator equal (if m is even) or with the degree of the denominator larger by one (if m is odd, cf. equation 3.2.2 above). For the derivation of the algorithm, refer to [1]. The algorithm is summarized by a recurrence relation exactly analogous to equation (3.1.3) for polynomial approximation:

$$R_{i(i+1)\dots(i+m)} = R_{(i+1)\dots(i+m)} + \frac{R_{(i+1)\dots(i+m)} - R_{i\dots(i+m-1)}}{\left(\frac{x-x_i}{x-x_{i+m}}\right)\left(1 - \frac{R_{(i+1)\dots(i+m)} - R_{i\dots(i+m-1)}}{R_{(i+1)\dots(i+m)} - R_{(i+1)\dots(i+m-1)}}\right) - 1} \tag{3.2.3}$$

This recurrence generates the rational functions through $m+1$ points from the ones through m and (the term $R_{(i+1)\dots(i+m-1)}$ in equation 3.2.3) $m-1$ points. It is started with

$$R_i = y_i \tag{3.2.4}$$

and with

$$R \equiv [R_{i(i+1)\dots(i+m)} \quad \text{with} \quad m = -1] = 0 \tag{3.2.5}$$

Now, exactly as in equations (3.1.4) and (3.1.5) above, we can convert the recurrence (3.2.3) to one involving only the small differences

$$\begin{aligned} C_{m,i} &\equiv R_{i\dots(i+m)} - R_{i\dots(i+m-1)} \\ D_{m,i} &\equiv R_{i\dots(i+m)} - R_{(i+1)\dots(i+m)} \end{aligned} \tag{3.2.6}$$

Note that these satisfy the relation

$$C_{m+1,i} - D_{m+1,i} = C_{m,i+1} - D_{m,i} \tag{3.2.7}$$

which is useful in proving the recurrences

$$\begin{aligned} D_{m+1,i} &= \frac{C_{m,i+1}(C_{m,i+1} - D_{m,i})}{\left(\frac{x-x_i}{x-x_{i+m+1}}\right) D_{m,i} - C_{m,i+1}} \\ C_{m+1,i} &= \frac{\left(\frac{x-x_i}{x-x_{i+m+1}}\right) D_{m,i}(C_{m,i+1} - D_{m,i})}{\left(\frac{x-x_i}{x-x_{i+m+1}}\right) D_{m,i} - C_{m,i+1}} \end{aligned} \tag{3.2.8}$$

This recurrence is implemented in the following function, whose use is analogous in every way to `polint` in §3.1. Note again that unit-offset input arrays are assumed (§1.2).

```
#include <math.h>
#include "nrutil.h"
#define TINY 1.0e-25                A small number.
#define FREERETURN {free_vector(d,1,n);free_vector(c,1,n);return;}

void ratint(float xa[], float ya[], int n, float x, float *y, float *dy)
Given arrays xa[1..n] and ya[1..n], and given a value of x, this routine returns a value of
y and an accuracy estimate dy. The value returned is that of the diagonal rational function,
evaluated at x, which passes through the n points (xa_i, ya_i), i = 1...n.
{
    int m,i,ns=1;
    float w,t,hh,h,dd,*c,*d;
```

```
    c=vector(1,n);
    d=vector(1,n);
    hh=fabs(x-xa[1]);
    for (i=1;i<=n;i++) {
        h=fabs(x-xa[i]);
        if (h == 0.0) {
            *y=ya[i];
            *dy=0.0;
            FREERETURN
        } else if (h < hh) {
            ns=i;
            hh=h;
        }
        c[i]=ya[i];
        d[i]=ya[i]+TINY;          The TINY part is needed to prevent a rare zero-over-zero
    }                                 condition.
    *y=ya[ns--];
    for (m=1;m<n;m++) {
        for (i=1;i<=n-m;i++) {
            w=c[i+1]-d[i];
            h=xa[i+m]-x;           h will never be zero, since this was tested in the initial-
            t=(xa[i]-x)*d[i]/h;        izing loop.
            dd=t-c[i+1];
            if (dd == 0.0) nrerror("Error in routine ratint");
            This error condition indicates that the interpolating function has a pole at the
            requested value of x.
            dd=w/dd;
            d[i]=c[i+1]*dd;
            c[i]=t*dd;
        }
        *y += (*dy=(2*ns < (n-m) ? c[ns+1] : d[ns--]));
    }
    FREERETURN
}
```


CITED REFERENCES AND FURTHER READING:

Stoer, J., and Bulirsch, R. 1980, *Introduction to Numerical Analysis* (New York: Springer-Verlag), §2.2. [1]

Gear, C.W. 1971, *Numerical Initial Value Problems in Ordinary Differential Equations* (Englewood Cliffs, NJ: Prentice-Hall), §6.2.

Cuyt, A., and Wuytack, L. 1987, *Nonlinear Methods in Numerical Analysis* (Amsterdam: North-Holland), Chapter 3.

3.3 Cubic Spline Interpolation

Given a tabulated function $y_i = y(x_i),\ i = 1...N$, focus attention on one particular interval, between x_j and x_{j+1}. Linear interpolation in that interval gives the interpolation formula

$$y = Ay_j + By_{j+1} \tag{3.3.1}$$