4.3 Romberg Integration

We can view Romberg's method as the natural generalization of the routine qsimp in the last section to integration schemes that are of higher order than Simpson's rule. The basic idea is to use the results from k successive refinements of the extended trapezoidal rule (implemented in trapzd) to remove all terms in the error series up to but not including $O(1/N^{2k})$. The routine qsimp is the case of $k = 2$. This is one example of a very general idea that goes by the name of *Richardson's deferred approach to the limit*: Perform some numerical algorithm for various values of a parameter h, and then extrapolate the result to the continuum limit $h = 0$.

Equation (4.2.4), which subtracts off the leading error term, is a special case of polynomial extrapolation. In the more general Romberg case, we can use Neville's algorithm (see §3.2) to extrapolate the successive refinements to zero stepsize. Neville's algorithm can in fact be coded very concisely within a Romberg integration routine. For clarity of the program, however, it seems better to do the extrapolation by a function call to Poly_interp::rawinterp, as given in §3.2.

romberg.h

```
template <class T>
Doub qromb(T &func, Doub a, Doub b, const Doub eps=1.0e-10) {
Returns the integral of the function or functor func from a to b. Integration is performed by
Romberg's method of order 2K, where, e.g., K=2 is Simpson's rule.
    const Int JMAX=20, JMAXP=JMAX+1, K=5;
    Here EPS is the fractional accuracy desired, as determined by the extrapolation error es-
    timate; JMAX limits the total number of steps; K is the number of points used in the
    extrapolation.
    VecDoub s(JMAX),h(JMAXP);            These store the successive trapezoidal approxi-
    Poly_interp polint(h,s,K);               mations and their relative stepsizes.
    h[0]=1.0;
    Trapzd<T> t(func,a,b);
    for (Int j=1;j<=JMAX;j++) {
        s[j-1]=t.next();
        if (j >= K) {
            Doub ss=polint.rawinterp(j-K,0.0);
            if (abs(polint.dy) <= eps*abs(ss)) return ss;
        }
        h[j]=0.25*h[j-1];
        This is a key step: The factor is 0.25 even though the stepsize is decreased by only
        0.5. This makes the extrapolation a polynomial in h^2 as allowed by equation (4.2.1),
        not just a polynomial in h.
    }
    throw("Too many steps in routine qromb");
}
```

The routine qromb is quite powerful for sufficiently smooth (e.g., analytic) integrands, integrated over intervals that contain no singularities, and where the endpoints are also nonsingular. qromb, in such circumstances, takes many, *many* fewer function evaluations than either of the routines in §4.2. For example, the integral

$$\int_0^2 x^4 \log(x + \sqrt{x^2 + 1})dx$$

converges (with parameters as shown above) on the second extrapolation, after just 6 calls to trapzd, while qsimp requires 11 calls (32 times as many evaluations of the integrand) and qtrap requires 19 calls (8192 times as many evaluations of the integrand).

CITED REFERENCES AND FURTHER READING:

Stoer, J., and Bulirsch, R. 2002, *Introduction to Numerical Analysis*, 3rd ed. (New York: Springer), §3.4 – §3.5.

Dahlquist, G., and Bjorck, A. 1974, *Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall); reprinted 2003 (New York: Dover), §7.4.1 – §7.4.2.

Ralston, A., and Rabinowitz, P. 1978, *A First Course in Numerical Analysis*, 2nd ed.; reprinted 2001 (New York: Dover), §4.10–2.

4.4 Improper Integrals

For our present purposes, an integral will be "improper" if it has any of the following problems:

- its integrand goes to a finite limiting value at finite upper and lower limits, but cannot be evaluated *right on* one of those limits (e.g., $\sin x/x$ at $x = 0$)
- its upper limit is ∞ , or its lower limit is $-\infty$
- it has an integrable singularity at either limit (e.g., $x^{-1/2}$ at $x = 0$)
- it has an integrable singularity at a known place between its upper and lower limits
- it has an integrable singularity at an unknown place between its upper and lower limits

If an integral is infinite (e.g., $\int_1^\infty x^{-1}dx$), or does not exist in a limiting sense (e.g., $\int_{-\infty}^{\infty} \cos x dx$), we do not call it improper; we call it impossible. No amount of clever algorithmics will return a meaningful answer to an ill-posed problem.

In this section we will generalize the techniques of the preceding two sections to cover the first four problems on the above list. A more advanced discussion of quadrature with integrable singularities occurs in Chapter 19, notably §19.3. The fifth problem, singularity at an unknown location, can really only be handled by the use of a variable stepsize differential equation integration routine, as will be given in Chapter 17, or an adaptive quadrature routine such as in §4.7.

We need a workhorse like the extended trapezoidal rule (equation 4.1.11), but one that is an *open* formula in the sense of §4.1, i.e., does not require the integrand to be evaluated at the endpoints. Equation (4.1.19), the extended midpoint rule, is the best choice. The reason is that (4.1.19) shares with (4.1.11) the "deep" property of having an error series that is entirely even in h. Indeed there is a formula, not as well known as it ought to be, called the *Second Euler-Maclaurin summation formula*,

$$\begin{aligned}\int_{x_0}^{x_{N-1}} f(x)dx = h[f_{1/2} + f_{3/2} + f_{5/2} + \cdots + f_{N-5/2} + f_{N-3/2}]\\ + \frac{B_2 h^2}{4}(f'_{N-1} - f'_0) + \cdots \\ + \frac{B_{2k} h^{2k}}{(2k)!}(1 - 2^{-2k+1})(f^{(2k-1)}_{N-1} - f^{(2k-1)}_0) + \cdots\end{aligned} \tag{4.4.1}$$

This equation can be derived by writing out (4.2.1) with stepsize h, then writing it out again with stepsize $h/2$, and then subtracting the first from twice the second.

It is not possible to double the number of steps in the extended midpoint rule and still have the benefit of previous function evaluations (try it!). However, it is