LECTURE 23: *Clenshaw–Curtis quadrature*

3.3 *Clenshaw–Curtis quadrature*

To get faster convergence for a fixed number of function evaluations, one might wish to increase the degree of the approximating polynomial further still, then integrate that high-degree polynomial. As evidenced in the discussion of polynomial interpolation in Section 1.6, the success of such an approach depends significantly on the choice of the interpolation points and the nature of the function being interpolated. For example, we would not expect the integral of a high degree polynomial interpolant to Runge's function $f(x) = (x^2+1)^{-1}$ over uniformly spaced points on $[-5,5]$ to accurately approximate

$$\int_{-5}^{5} \frac{1}{x^2+1}\,\mathrm{d}x,$$

since the underlying interpolants fail to converge for this example.

However, recall that Theorem 2.6 ensures that the polynomials that interpolate f *at Chebyshev points* will converge, provided f is just a bit smooth. This suggests that the integrals of such interpolating polynomials will also be accurate as n increases, and indeed this is the case.

This procedure of *integrating interpolants at Chebyshev points* is known as *Clenshaw–Curtis quadrature*. If f is smooth, this method typically converges much faster than the composite trapezoid and Simpson's rules, which are only based on low-degree polynomial interpolants. In fact, the Clenshaw–Curtis method competes very well with the Gaussian quadrature schemes discussed in the next sections, although those Gaussian quadrature schemes have historically received much greater attention.

See L. N. Trefethen, 'Is Gauss Quadrature Better than Clenshaw–Curtis?', *SIAM Review* 50 (2008) 67–87.

Suppose we wish to integrate a function over $[-1,1]$. Then Clenshaw–Curtis quadrature evaluates f at the $n+1$ Chebyshev points

$$x_j = \cos\left(\frac{j\pi}{n}\right), \quad j = 0,\ldots,n.$$

The Lagrange form of the polynomial interpolant at these points takes the form

$$p_n(x) = \sum_{j=0}^{n} f(x_j)\ell_j(x),$$

where

$$\ell_j(x) = \prod_{k=0}^{n} \frac{x-x_k}{x_j-x_k}$$

are the usual Lagrange basis functions. Since the Clenshaw–Curtis quadrature rule will integrate the polynomial interpolant at the Chebyshev points, the rule will give

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \int_{-1}^{1} p_n(x)\,\mathrm{d}x = \sum_{j=0}^{n} f(x_j) \int_{-1}^{1} \ell_j(x)\,\mathrm{d}x.$$

Thus, defining the *weights* to be

$$w_j := \int_{-1}^{1} \ell_j(x)\,\mathrm{d}x,$$

the Clenshaw–Curtis quadrature rule takes the following compact form.

Clenshaw–Curtis rule:

$$\int_{-1}^{1} f(x)\,\mathrm{d}x \approx \sum_{j=0}^{n} w_j f(x_j),$$

where $x_j = \cos(j\pi/n)$ and $w_j = \int_{-1}^{1} \ell_j(x)\,\mathrm{d}x$.

Connecting interpolation at Chebyshev points to trigonometric interpolation leads to a convenient algorithm for computing the weights w_j using a fast Fourier transform, which is much more stable and convenient than integrating the Lagrange interpolating polynomials directly. We shall not go into details here. Interested readers can consult Trefethen's paper, 'Is Gauss Quadrature Better than Clenshaw–Curtis?' (2008) or his book *Spectral Methods in MATLAB* (2000).