Numerical integration: Newton–Cotes quadrature rules

In the previous class we introduced Newton–Cotes quadrature rules, whereby an approximate value of the definite integral $\int_a^b f(x)\,dx$ is obtained by integrating the Lagrange polynomial that interpolates f at n equally spaced nodes $x_1, \ldots, x_n$ in the interval $[a, b]$. There are two types of Newton–Cotes rules: *closed* rules use the endpoints a and b as nodes, so the horizontal distance from one node to the next (the "step size") is $h = \dfrac{b-a}{n-1}$; *open* rules do not include the endpoints as nodes, i.e. the n nodes are all inside the open interval (a, b), with the horizontal distance from one node to the next being $h = \dfrac{b-a}{n+1}$.

Closed Newton–Cotes formulas

$\boxed{n = 2}$ Trapezoidal rule $\qquad x_1 = a, \qquad x_2 = b, \qquad h = b - a$

$$\frac{1}{2}h\big[f(x_1) + f(x_2)\big]$$

$\boxed{n = 3}$ Simpson's rule

$\boxed{n = 4}$ Simpson's 3/8 rule

$\boxed{n = 5}$ Boole's rule

Open Newton–Cotes formulas

$\boxed{n = 1}$ Midpoint rule $\qquad x_1 = \dfrac{a+b}{2}, \qquad h = \dfrac{b-a}{2}$

$$2hf(x_1)$$

$\boxed{n = 2}$

$\boxed{n = 3}$

$\boxed{n = 4}$

Composite quadrature rules

Using a single Newton–Cotes formula to compute an integral does not give good results unless the integration interval is quite small. Over a larger interval, in order to approximate the integrand f reasonably well one would need a greater number of interpolation nodes and a higher-degree polynomial, but then polynomial wiggle (Runge's phenomenon) could become a problem. Instead, we take a *piecewise* approach: divide the original interval $[a, b]$ into subintervals and apply a Newton–Cotes formula to each subinterval.

When using a computer to perform numerical integration, most of the work lies in evaluating the function at the nodes. Therefore, to increase computational efficiency, one should try to subdivide the interval in such a way that previously computed function values can be reused. In writing programs to implement composite numerical integration, the usual practice is to subdivide the interval in a recursive manner, e.g. divide $[a, b]$ in two, then divide each half in two, and so on—this bisection strategy would be good for the trapezoidal rule, Simpson's rule, Boole's rule, and 3-point open Newton–Cotes rule, for instance.

We shall write a function that implements composite Simpson's rule, computing approximations of $\int_a^b f(x)\,dx$ to a given accuracy (tolerance) by repeatedly halving h. The program will also keep track of the number of function evaluations done.