1 Numerical integration

Numerical integration, also called *quadrature*, is an algorithm to compute an approximation to a definite integral in the form of a finite sum,

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \,, \tag{1}$$

where the abscissas x_i and the weights w_i are chosen such that the quadrature is particularly well suited for a given problem. Different quadratures use different strategies of choosing the abscissas and weights.

1.1 Classical quadratures with equally spaced abscissas

Classical quadratures use predefined, often equally-spaced, abscissas, e.g.

$$x_i = \frac{i-1}{n-1}(b-a) \,,\ i = 1, \ldots, n \,. \tag{2}$$

A quadrature is called *closed* if the abscissas include the end-points of the interval or the mid-point (which becomes end-point after halving the interval). Otherwise it is called *open*. If the integrand is diverging at the end-points (or at the mid-point of the interval) the close quadratures generally can not be used.

For an n-point classical quadrature the n free parameters w_i can be chosen such that the quadrature integrates exactly a set of n (linearly independent) functions $\{f_1(x), \ldots, f_n(x)\}$ where the integrals

$$I_k \equiv \int_a^b f_k(x)dx \tag{3}$$

are known. This gives a set of equations, linear in w_i,

$$\sum_{i=1}^{n} w_i f_k(x_i) = I_k \,, \tag{4}$$

where $k = 1, \ldots, n$. The weights w_i can be easily determined by solving the linear system (4).

If the functions to be integrated exactly are chosen as polynomials $\{1, x, x^2, \ldots, x^{n-1}\}$, the quadrature is called Newton-Cotes quadrature. The n-point Newton-Cotes quadrature can integrate exactly the first n terms of the function's Taylor expansion[1]

$$f(a+t) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!} t^k \,. \tag{6}$$

[1] assuming that the integral is rescaled as

$$\int_a^b f(x)dx = \int_0^{h=b-a} f(a+t)dt \,. \tag{5}$$

The nth order term $\frac{f^{(n)}(a)}{n!}t^n$ will not be integrated exactly by an n-point quadrature and will then result in the quadrature's error[2]

$$\epsilon_n \approx \int_0^h \frac{f^{(n)}(a)}{n!} t^n dt = \frac{f^{(n)}(a)}{n!(n+1)} h^{n+1} \,. \tag{7}$$

If the function is smooth and the interval h is small enough the Newton-Cotes quadrature can give a good approximation.

Here are a few examples of classical quadratures: one-point closed,

$$\int_0^h f(x)dx \approx h f(\tfrac{1}{2}h) \,, \tag{8}$$

three-point closed,

$$\int_0^h f(x)dx \approx \tfrac{1}{6}h \left(f(0) + 4f(\tfrac{1}{2}h) + f(h) \right) \,, \tag{9}$$

two-point open,

$$\int_0^h f(x)dx \approx \tfrac{1}{2}h \left(f(\tfrac{1}{3}h) + f(\tfrac{2}{3}h) \right) \,, \tag{10}$$

four-point open,

$$\int_0^h f(x)dx \approx \tfrac{1}{6}h \begin{pmatrix} 2f(\frac{1}{6}h) + f(\frac{2}{6}h) \\ +f(\frac{4}{6}h) + 2f(\frac{5}{6}h) \end{pmatrix} \,. \tag{11}$$

1.2 Quadratures with optimal abscissas

In quadratures with optimal abscissas, called *Gaussian quadratures*, not only weights w_i but also abscissas x_i are chosen optimally. The number of free parameters is thus $2n$ (n abscissas and n weights) and one can chose $2n$ functions $\{f_1(x), \ldots, f_{2n}(x)\}$ to be integrated exactly. This gives a system of $2n$ equations, linear in w_i and non-linear in x_i,

$$\sum_{i=1}^{n} w_i f_k(x_i) = I_k \,, \tag{12}$$

where $k = 1, \ldots, 2n$, and $I_k = \int_a^b f_k(x)dx$, from which the weights and abcissas can be determined.

The Gaussian quadratures are of order $2n-1$ compared to only order $n-1$ for non-optimal abscissas. However, the optimal points generally can not be reused at the next iteration in an adaptive algorithm.

[2] Actually the error is often one order in h higher due to symmetry of the the polynomials t^k.

Here is, for example, a two-point Gauss-Legendre quadrature rule [3]

$$\int_{-1}^{1} f(x)dx \approx f\left(-\sqrt{\tfrac{1}{3}}\right) + f\left(+\sqrt{\tfrac{1}{3}}\right) . \quad (14)$$

1.3 Subdivision of the interval vs higher order quadrature

The higher order quadratures, say $n > 10$, suffer from round-off errors as the weights w_i generally have alternating signs. Again, using high order polynomials is dangerous as they typically oscillate wildly and may lead to Runge phenomenon. Therefore if the error of the quadrature is yet too big for a sufficiently large n quadrature, the best strategy is to subdivide the interval in two and then use the quadrature on the half-intervals. Indeed, if the error is of the order h^k, the subdivision would lead to reduced error, $2\left(\frac{h}{2}\right)^k < h^k$, if $k > 1$.

1.4 Adaptive quadratures

Adaptive quadrature is an algorithm where the integration interval is subdivided into adaptively refined subintervals until the given accuracy goal is reached.

Adaptive algorithms are usually built on pairs of quadrature rules, a higher order rule (eg open-4) and a lower order rule (eg open-2). The higher order rule is used to compute the approximation, Q, to the integral. The difference between the higher order rule and the lower order rule gives an estimate of the error, δQ. If the integration result is accepted, if

$$\delta Q < \delta + \epsilon|Q| . \quad (15)$$

where δ is the absolute accuracy goal and ϵ is the relative accuracy goal.

Otherwise the interval is subdivided into two half- intervals and the procedure applies recursively to the subintervals with the same relative accuracy goal ϵ and rescaled absolute accuracy goal $\delta/\sqrt{2}$.

The reuse of the function evaluations made at the previous step is very important for the efficiency of the algorithm. The equally-spaced abscissas naturally provide such a reuse.

1.5 Gauss-Kronrod quadratures

Gauss-Kronrod quadratures represent a compromise between equally spaced abscissas and optimal abscissas: n points are reused from the previous iteration (n weights as free parameters) and then m optimal points are added (m abscissas and m weights as free parameters). Thus the accuracy of the method is $n + 2m - 1$. There are several special variants of these quadratures fit for particular types of the integrands.

[3] assuming that the integral is rescaled as

$$\int_a^b f(x)dx = \int_{-1}^{1} \frac{b-a}{2} f\left(\frac{a+b}{2} + \frac{b-a}{2}t\right) dt . \quad (13)$$

Table 1: Recursive adaptive integrator based on open-2/4 quadratures.

```
function adapt24(f,a,b,acc,eps,oldfs){
var x=[1/6,2/6,4/6,5/6];// abscissas
var w=[2/6,1/6,1/6,2/6];// high order
var p=[1,0,0,1];// which points are new
var v=[1/4,1/4,1/4,1/4];// low order
var n=x.length, h=b-a;
if(typeof(oldfs)=="undefined")// first call
  fs=[f(a+x[i]*h) for(i in x)];
else for(let k=0,i=0;i<n;i++){
  if(p[i]) fs[i]=f(a+x[i]*h);
  else     fs[i]=oldfs[k++];}
for(var q4=q2=i=0;i<n;i++){
  q4+=w[i]*fs[i]*h;
  q2+=v[i]*fs[i]*h;}
var tol=acc+eps*Math.abs(q4)
var err=Math.abs(q4-q2)/7
if(err<tol)
  return [q4, err]
else{
  acc/=Math.sqrt(2.)
  var mid=(a+b)/2
  var left=[fs[i]for(i in fs)if(i< n/2)]
  var rght=[fs[i]for(i in fs)if(i>=n/2)]
  var [ql,el]=adapt24(f,a,mid,eps,acc,left)
  var [qr,er]=adapt24(f,mid,b,eps,acc,rght)
  return [ql+qr, Math.sqrt(el*el+er*er)]
  }
}
```