Numerical integration

Numerical integration, also called *quadrature* for one-dimensional integrals and *cubature* for multi-dimensional integrals, is an algorithm to compute an approximation to a definite integral in the form of a finite sum,

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

The abscissas x_i and the weights w_i in (1) are chosen such that the quadrature is particularly well suited for the given class class of functions to integrate. Different quadratures use different strategies of choosing the abscissas and weights.

Classical quadratures with equally spaced abscissas

Classical quadratures use predefined equally-spaced abscissas. 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 closed 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 $\{\phi_1(x), \dots, \phi_n(x)\}$ where the integrals

$$I_k \equiv \int_a^b \phi_k(x)dx \tag{2}$$

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

$$\sum_{i=1}^{n} w_i \phi_k(x_i) = I_k \; , \; k = 1 \dots n \; . \tag{3}$$

The weights w_i can then be determined by solving the linear system (3).

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

$$f(a+t) = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!} t^k \; . \tag{4}$$

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[1] in the quadrature's error[2]

$$\epsilon_n \approx \int_0^h \frac{f^{(n)}(a)}{n!} t^n dt = \frac{f^{(n)}(a)}{n!(n+1)} h^{n+1} \; . \tag{6}$$

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

[1] Assuming that the integral is rescaled as

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

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

Here are several examples of closed and open classical quadratures:

$$\int_0^h f(x)dx \approx \tfrac{1}{2}h\left[f(0)+f(h)\right] , \tag{7}$$

$$\int_0^h f(x)dx \approx \tfrac{1}{6}h\left[f(0)+4f(\tfrac{1}{2}h)+f(h)\right] , \tag{8}$$

$$\int_0^h f(x)dx \approx \tfrac{1}{2}h\left[f(\tfrac{1}{3}h)+f(\tfrac{2}{3}h)\right] , \tag{9}$$

$$\int_0^h f(x)dx \approx \tfrac{1}{6}h\left[2f(\tfrac{1}{6}h)+f(\tfrac{2}{6}h)+f(\tfrac{4}{6}h)+2f(\tfrac{5}{6}h)\right] . \tag{10}$$

Quadratures with optimized 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 optimal abscissas and n weights) and one can chose $2n$ functions $\{\phi_1(x),\dots,\phi_{2n}(x)\}$ to be integrated exactly. This gives a system of $2n$ equations, linear in w_i and non-linear in x_i,

$$\sum_{i=1}^{n} w_i f_k(x_i) = I_k \; , k = 1,\dots,2n, \tag{11}$$

where $I_k = \int_a^b f_k(x)dx$. The weights and abcissas can be determined by solving this system of equations.

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

$$\int_{-1}^{1} f(x)dx \approx f\left(-\sqrt{\tfrac{1}{3}}\right) + f\left(+\sqrt{\tfrac{1}{3}}\right) . \tag{13}$$

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

Reducing the error by subdividing the interval

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 $k > 1$.

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 (preferably using the same points), a higher order rule (e.g. 4-point-open) and a lower order rule (e.g. 2-point-open). 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. The integration result is accepted, if

$$\delta Q < \delta + \epsilon|Q| , \tag{14}$$

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

[3] assuming that the integral is rescaled as

$$\int_a^b f(x)dx = \int_{-1}^{1} \frac{b-a}{2} f\left(\frac{a+b}{2} + \frac{b-a}{2}t\right) dt . \tag{12}$$

Otherwise the interval is subdivided into two half-intervals and the procedure applies recursively to 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 of adaptive integration is very important for the efficiency of the algorithm. The equally-spaced abscissas naturally provide for such a reuse.

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

```
function adapt(f,a,b,acc,eps,oldfs){// adaptive integrator

var x=[1/6,2/6,4/6,5/6];// abscissas
var w=[2/6,1/6,1/6,2/6];// weights of higher order quadrature
var v=[1/4,1/4,1/4,1/4];// weights of lower order quadrature
var p=[1,0,0,1];// shows the new points at each recursion
var n=x.length, h=b-a;

if(typeof(oldfs)=="undefined") // first call?
  fs=[f(a+x[i]*h) for(i in x)]; //first call: populate oldfs
else{ // recursive call: oldfs are given
  fs = new Array(n);
  for(var k=0,i=0;i<n;i++){
    if(p[i]) fs[i]=f(a+x[i]*h); // new points
    else     fs[i]=oldfs[k++];}} // reuse of old points

for(var q4=q2=i=0;i<n;i++){
  q4+=w[i]*fs[i]*h; // higher order estimate
  q2+=v[i]*fs[i]*h;} // lower order estimate
var tol=acc+eps*Math.abs(q4) // required tolerance
var err=Math.abs(q4-q2)/3 // error estimate

if(err<tol) // are we done?
  return [q4, err] // yes, return integral and error
else{ // too big error, preparing the recursion
  acc/=Math.sqrt(2.) // rescale the absolute accuracy goal
  var mid=(a+b)/2
  var left=[fs[i]for(i in fs)if(i< n/2)] // store the left points
  var rght=[fs[i]for(i in fs)if(i>=n/2)] // store the right points
  var [ql,el]=adapt(f,a,mid,eps,acc,left) // dispatch two recursive calls
  var [qr,er]=adapt(f,mid,b,eps,acc,rght)
  return [ql+qr, Math.sqrt(el*el+er*er)] // return the grand estimates
  }
}
```

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.

Integrals over infinite intervals

Infinite intervals

One way to calculate an integral over infinite interval is to transform it into an integral over a finite interval,

$$\int_{-\infty}^{+\infty} f(x)dx = \int_{-1}^{+1} f\left(\frac{t}{1-t^2}\right)\frac{1+t^2}{(1-t^2)^2}dt \;, \tag{15}$$

by the variable substitution

$$x = \frac{t}{1-t^2} \ , \ dx = \frac{1+t^2}{(1-t^2)^2} dt \ , \ t = \frac{\sqrt{1+4x^2}-1}{2x} \ . \tag{16}$$

The integral over finite interval can be evaluated by ordinary integration methods.

Alternatively,

$$\int_{-\infty}^{+\infty} f(x)dx = \int_0^1 \frac{dt}{t^2}\left(f\left(\frac{1-t}{t}\right) + f\left(-\frac{1-t}{t}\right)\right) \ . \tag{17}$$

Half-infinite intervals

An integral over a half-infinite interval can be transformed into an integral over a finite interval,

$$\int_a^{+\infty} f(x)dx = \int_0^1 f\left(a + \frac{t}{1-t}\right) \frac{1}{(1-t)^2} dt \ , \tag{18}$$

by the variable substitution

$$x = a + \frac{t}{1-t} \ , \ dx = \frac{1}{(1-t)^2} dt \ , \ t = \frac{x-a}{1+(x-a)} \ . \tag{19}$$

Similarly,

$$\int_{-\infty}^{a} f(x)dx = \int_{-1}^{0} f\left(a - \frac{t}{1+t}\right) \frac{-1}{(1+t)^2} dt \ , \tag{20}$$

Alternatively,

$$\int_a^{+\infty} f(x)dx = \int_0^1 f\left(a + \frac{1-t}{t}\right) \frac{dt}{t^2} \tag{21}$$

$$\int_{-\infty}^{b} f(x)dx = \int_0^1 f\left(b - \frac{1-t}{t}\right) \frac{dt}{t^2} \tag{22}$$