1 Monte Carlo quadratures

Monte Carlo integration is a numerical quadrature where the abscissas are chosen randomly and no assumptions about smoothness of the integrand are made, not even that the integrand is continuous.

Plain Monte Carlo algorithm distributes points (in a process called "sampling") uniformly from the integration region using either uncorrelated pseudo-random or correlated quasi-random sequences of points.

Adaptive algorithms, such as VEGAS and MISER, distribute points non-uniformly, attempting to reduce integration error, using "importance" and "stratified" sampling, correspondingly.

1.1 Multi-dimensional integration

One of the problems in multi-dimensional integration is that often the integration region Ω is quite complicated, with the boundary not easily described by simple functions. However it is usually a lot easier to find out whether a given points lies within the integration region or not. Therefore a popular strategy is to create an auxiliary rectangular volume V which contains the integration volume Ω an an auxiliary function F which coincides with the integrand inside the volume Ω and is equal zero outside. Then the integral of the auxiliary function over the (simple rectangular) auxiliary volume is equal the original integral.

Unfortunately the auxiliary function is non-continuous at the boundary and thus the ordinary quadratures which assume continuous integrand will fail badly here while the Monte Carlo quadratures will do just as good (or bad) as with continuous integrand.

1.2 Plain Monte Carlo sampling

Plain Monte Carlo is a quadrature with equal weights and non-optimised random abscissas,

$$\int_V f(\mathbf{x})dV \approx w\sum_{i=1}^{N} f(\mathbf{x}_i)\ , \tag{1}$$

where $\mathbf{x}$ a point in the multi-dimensional integration space. One free parameter, w, allows one condition to be satisfied: the quadrature has to integrate exactly a constant function. This gives $w = V/N$,

$$\int_V f(\mathbf{x})dV \approx \frac{V}{N}\sum_{i=1}^{N} f(\mathbf{x}_i) = V\langle f\rangle\ . \tag{2}$$

According to the *central limit theorem* the error estimate ϵ is close to

$$\epsilon = V\frac{\sigma}{\sqrt{N}}\ , \tag{3}$$

where σ is the variance of the sample,

$$\sigma^2 = \langle f^2\rangle - \langle f\rangle^2. \tag{4}$$

The $1/\sqrt{N}$ convergence of the error is quite slow.

Table 1: Plain Monte Carlo integrator

```
function plainmc(fun, a, b, N) {
// integrates function "fun" over
// a rectangular volume defined by
// the lower-left point "a" and
// the upper-right point "b"
// using N pseudo-random points;
// returns estimates of the integral
// and the integration error
var randomx = function(a,b)
   [a[i]+Math.random()*(b[i]-a[i])
     for (i in a)];
var V=1; for(var i in a) V*=b[i]-a[i];
var sum=0,sum2=0;
for(var i=0;i<N;i++){
  var f=fun(randomx(a,b));
  sum+=f; sum2+=f*f}
var average =sum/N;
var variance=sum2/N-average*average;
var Q=V*average;
var err=V*Math.sqrt(variance/N);
return [Q,err];
}
```

1.3 Importance sampling

Suppose that the points are distributed not uniformly but with some density $\rho(x)$: the number of points Δn in the volume ΔV around point x is given as

$$\Delta n = \frac{N}{V}\rho\Delta V, \tag{5}$$

where ρ is normalised as $\int_V \rho dV = V$.

The estimate of the integral is then given as

$$\int_V f(\mathbf{x})dV \approx \sum_{i=1}^{N} f(x_i)\Delta V_i = V\left\langle\frac{f}{\rho}\right\rangle\ , \tag{6}$$

where $\Delta V_i = \frac{V}{N\rho(x_i)}$ is the 'volume per point' at the point x_i.

The corresponding variance is now

$$\sigma^2 = \left\langle\left(\frac{f}{\rho}\right)^2\right\rangle - \left\langle\frac{f}{\rho}\right\rangle^2\ , \tag{7}$$

Apparently if the ration f/ρ is close to a constant, the variance is reduced. It is tempting to take $\rho = |f|$ and sample directly from the function. However in practice it is typically expensive to evaluate the integrand. Therefore a better strategy is to build an approximate density in the product form, $\rho(x, y, \ldots, z) = \rho_x(x)\rho_y(y)\ldots\rho_z(z)$, and then sample from this approximate density. A popular routine of this sort is called VEGAS. The sampling from a given function can be done using the Metropolis algorithm which we shall not discuss here.

1.4 Stratified sampling

Stratified sampling is a generalisation of the recursive adaptive integration algorithm to random quadratures in multi-dimensional spaces.

The ordinary 'dividing by two' strategy does not work for multi-dimensions as the number of sub-volumes grows way too fast to keep track of. Instead one estimates along which dimension a subdivision should bring the most dividends and only subdivides along this dimension. The method is called *recursive stratified sampling*. A typical algorithm is listed in Table 2.

Table 2: Recursive stratified sampling

```
Sample $N$ random points;
Estimate the average and the error;
If the error is acceptable :
    Return the average and the error;
Else :
    For each dimension :
        Subdivide the volume in two along the dimension;
        Estimate the sub-averages in the two sub-volumes;
    Pick the dimension with the largest sub-average;
    Subdivide the volume in two along this dimension;
    Dispatch two recursive calls to each of the sub-volumes;
    Estimate the grand average and grand error;
    Return the grand average and grand error;
```

In a stratified sample the points are concentrated in the regions where the variance of the function is largest, as illustrated on Figure 1.

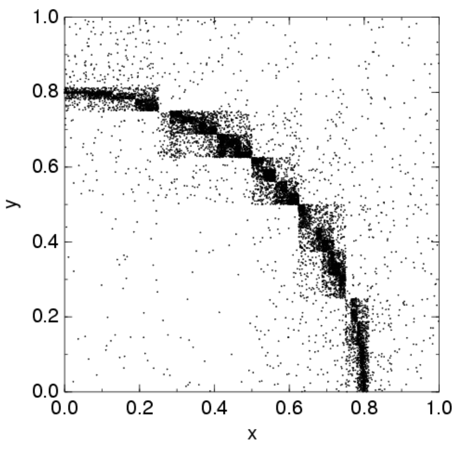

Figure 1: Stratified sample of a discontinuous function $f(x, y) = (\sqrt{x^2 + y^2} < 0.8)\ ?\ 1 : 0$