Topic 10
Monte Carlo Methods

References

R Landau et al, *Computational Physics - Problem Solving with Computers* parts available at http://www.physics.orst.edu/ rubin/CPbook/

D. Knuth, *The Art of Computer Programming*, Addison-Wesley

Introduction

Thus far, we have learned of three numerical intergration methods: Euler's Method, Euler's Improved Method, and the Runge Kutta method.

We learned that each method evaluates the integrand up to 4 times.

Method	#Equations	#Evals
Eul	$y_{new} = y + c_1$	1
EulImp	$y_{new} = y + \frac{1}{2}(c_1 + c_2)$	2
RK	$y_{new} = y + \frac{1}{6}(c_1 + 2c_2 + 2c_3 + c_4)$	4

If we have many integration steps to perform over multiple dimensions, this can get very computationally expensive.

There are several alternative methods of effectively achieving a numerical integration. They form a subset of the general technique known as *Monte Carlo* methods/simulations/techniques.

Monte Carlo (MC) Simulations

Essentially any techniques involving the use of random numbers in physical or mathematical problems can be called an MC method. The name "Monte Carlo" was coined by N. Metropolis (inspired by Stanley Ulam's interest in poker) during the Manhattan Project of World War II, because of the similarity of statistical simulation to games of chance.

However, MC methods have been used for centuries, but only in the past several decades has the technique gained the status of a full-fledged numerical method capable of addressing the most complex applications.

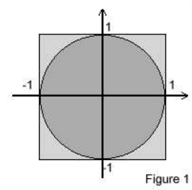
In many applications of Monte Carlo, the physical process is simulated directly, and there is no need to even write down the differential equations that describe the behavior of the system. The only requirement is that the physical (or mathematical) system be described by probability density functions (pdf's),

Uses of the MC method

The MC method is used in

- Nuclear Reactor Design
- Quantum Dynamics
- Radiation Cancer Therapy
- Traffic Flow
- Stellar Evolution
- Econometrics
- Dow-Jones / All-ords forecasting
- Oil-well Exploration
- VLSI Design

Example: Integration by Rejection



Consider the unit circle (radius=1) within a square with sides equal to 2 (see figure 1). If we pick a random point (x,y) were both x and y are between $-1, \ldots, 1$, the probability that this random point lies inside the unit circle is given as the proportion between the area A, of the unit circle and the square:

$$P(x^2 + y^2 < 1) = \frac{A_{circle}}{A_{square}} = \frac{\pi}{4}$$

If we pick a random point N times and M of those times the point lies inside the unit circle, the probability of that a random point lies inside the unit circle is given as:

$$P(x^2 + y^2 < 1) = \frac{M}{N}$$

But if N becomes very large (theoretically infinite), the two probabilities will tend to become equal, so we can write:

$$\frac{4M}{N} \to \pi, \quad \text{as } N \to \infty$$

This C code calculates π using the MC method.

```
int n = 0, m = 0; double x,y;

for (n = 0; n < TOTALNUM; n++) {
  x = 2.0*rand() - 1.0;
  y = 2.0*rand() - 1.0;
  if ((x*x)+(y*y) < 1) n++;
}
pi = 4.0*m/n;
```

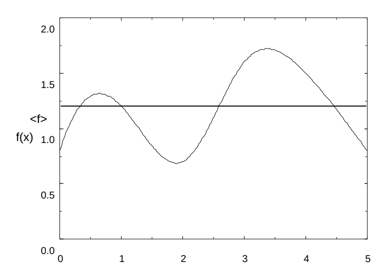
In the final formula for π, the precision (number of digits) depends on how many times you pick a point. The greater N and M, the more digits precision you get. However, the discrete representation of real numbers in a computer is also a limiting factor for the precision of this method. Because we only have a finite number of points (10^{19} for 32 bits) to choose from, we actually approximate a rational number close to pi, rather than pi itself!

The above program, run this program 10 times with `TOTALNUM = 100000` on a simple i386 based machine gave the value 3.14142 ± 0.00441 - not bad!

In some cases it is much easier to perform a rejection test (eg. linear-programming with constraints) than to perform the integration.

Integration by Mean Value

When an integration process covers many particles over many dimensions with a small step size, the complexity of the problem can blow up quickly. The figure below shows that we can approximate and integration by a mean value.



From the *Mean Value Theorem*,

$$I = \int_a^b dx \quad f(x) = (b - a)\langle f \rangle$$

where $\langle f \rangle$ represents the average value of f over the interval.

We can determine the sample mean by sampling the function at discrete intervals x_i so that

$$\bar{f} = \langle f \rangle \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

The uncertainty in the value obtained for the integral I after N samples of $f(x_i)$ is measured by the standard deviation σ_I. The standard deviation of the integrand f in the sampling is an intrinsic property of the function $f(x)$.

For normal distributions, they are related by

$$\sigma_I = \frac{1}{\sqrt{N}} \sigma_f.$$

Thus for large N, the error decreases as $\frac{1}{\sqrt{N}}$.

For higher dimensions this scales naturally, so that

$$\int_a^b dx \int_c^d dy \; f(x,y) = (b-a)(d-c) \frac{1}{N} \sum_{i=1}^{N} f(x_i) = (b-a)(d-c)\langle f \rangle$$

This significantly simplifies the integration step.

Error Behaviour

By the mean value theorem approximations, the error decreases as $\frac{1}{\sqrt{N}}$ for large N. If we spread N sample points over D dimensions, we have N/D points per dimension. The number of points per integral thus decreases as D increases, thus the error in each integration also increases with D.

This means that, using the Monte Carlo method, for every doubling of N we get 1.4 times greater accuracy,

For $D \approx 3 - 4$, Monte Carlo performs similarly to conventional integrations. For higher D, the Monte Carlo method is superior.