Lec9

- Numerical integration:direct
- (Pseudo)Random numbers
- Monte Carlo integration

Important numerical algorithms

- We have learnt how to approximate derivatives and to solve simple differential equations

- Another important task: integration.

- Like derivatives 2 types of problem. Want essentially exact solution to problem with few variables. Or wanting a good estimate (with errors) of a complex problem with very many variables

Simple integrals

Consider the following integral

$$I = \int_a^b dx 4\sqrt{1 - x^2}$$

Simplest approach:
Evaluate as area using slices (calculus definition):

$$I \sim \lim_{N \to \infty} \delta x \sum_{i=1}^{N} f(x_i)$$

$x_i = i\delta x$ and $\delta x = (b - a)/N$
Assumes $f(x)$ constant over (small) interval δx

Errors

Consider area of one slice running from $x = 0$ to $x = \delta x$.
Let $x = 0 + y$
Taylor expand:
$f(x) = f(0 + y) = f(0) + yf'(0) + \ldots$

$$\int_0^{\delta x} dx f(x) = \int_0^{\delta x} dy \left(f(0) + yf'(0)y + \ldots \right)$$

Simple slicing result gotten by taking first term.
Error $O(\delta x)$
Need small δx for good answer. How small ?
Need $\delta x << f'(0)$.

Classical algorithms

Pros:

Can be improved - Trapezoidal rule - take 2nd term:

$$\int_0^\delta dy(f(0) + yf'(0)) = \frac{1}{2}(f(0) + f(\delta x))$$

since $f' = (f(\delta x) - f(0))/\delta x$

Summing over all slices:

$$I \sim \frac{b-1}{2N} \sum_{i=1}^{N} (f(a) + 2f(x + \delta x) + \ldots f(b))$$

error: $O(\delta x^2)$.

Simpson's rule error $O(\delta x^3)$ arises by approximating $f(x)$ by quadratic function within slice.

Problems

- For many variables (dimensions) error decreases slowly.

- Need to make grid in the integration space with spacing δx

- If $f(x) \rightarrow f(x_1, \ldots x_d)$ and N function evaluations error goes as $O(N^{-\frac{1}{d}})$ – slow!

- Non-trivial integration regions difficult.

Simple Monte Carlo

Approx integral by:

$$I \sim \lim_{N\to\infty} \frac{b-a}{N} \sum_{i=1}^{N} f(x_i)$$

where now points x_i *are chosen at random*
Derive: mean value theorem of calculus
Error: $O(N^{-\frac{1}{2}})$ *independent of number of variables!!*
Conclusion: many variables Monte Carlo methods are essential.

Aside: random numbers

- Most computers/languages provide a simple *pseudorandom* number generator eg. `random()` in python.

- Often use *deterministic* algorithm to generate sequence:
  $r_{n+1} = (ar_n + b) \bmod m$

- random number in range $0, 1$ is just r_n/m. Careful choices of a, b, m generate long sequences of seemingly random numbers

- But such generators *will* repeat eventually. Care must be taken. In principle should check all results with at least 2 different random number generators.

Errors in Monte Carlo

Clearly value of integral depends on exact random numbers used.
Make several (eg 10) independent Monte Carlo measurements of the integral and look at how the different estimates of I_i differ from one another.
The error is clearly related to the standard deviation of these estimates

$$\delta I = \sqrt{\frac{1}{N}\sum_{i=1}^{N} < f_i^2 > - < f_i >^2}$$

Furthermore we can show that if we average the N measurements of I_i to yield I the error in this is just

$$\delta I \sim N^{-\frac{1}{2}}$$

as we increase the number of Monte Carlo runs.
Compare with error from simple slicing.

Importance Sampling

Simple uniform sampling in interval $(0, 1)$ not optimal.
Better to try to pick points x_i where function is largest
Rewrite integral as follows

$$I = \int f(x)dx = \int \frac{f(x)}{p(x)} p(x)dx$$

where $p(x)$ is some arbitrary probability distribution chosen so that the *variance* of $\frac{f(x)}{p(x)}$ is small
Then

$$f \sim \frac{b-a}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}$$

and points x_i drawn from probability distribution $p(x)$.

Metropolis algorithm

Want to compute

$$< f >= \frac{\int p(x) f(x) dx}{\int p(x) dx}$$

where $p(x)$ arbitrary (unnormalized) probability.

Metropolis method produces random walk of points x_i whose distribution approaches $p(x)$ after a large number of steps.

Transition probability $t(x_i \to x_j)$ satisfies *detailed balance* condition

$$p(x_i) t(x_i \to x_j) = p(x_j) t(x_j \to x_i)$$

One solution:

$$t(x_i \to x_j) = \mathrm{min}[1, \frac{p(x_j)}{p(x_i)}]$$

Implementation

- Choose trial position $x_{trial} = x_{old} + \delta$
- Calculate $w = p(x_{trial})/p(x_{old})$
- If $w > 1$ accept x_{trial}
- If $w < 1$ generate random number r
- If $r \leq w$ accept change
- If not $x_{trial} = x_{old}$

Choose δ so that 50% moves accepted approx.

Example

Consider

$$I = \int_0^1 x\sqrt{1-x^2}$$

Take $p(x) = \sqrt{1-x^2}$ Use Metropolis to estimate integral. Notice convergence of error like $N^{-\frac{1}{2}}$