Class 13. Numerical Integration

Simple Monte Carlo Integration (*NRiC* §7.6)

- Can use RNGs to estimate integrals.
- Suppose we pick n random points $\mathbf{x}_1$, ..., $\mathbf{x}_N$ uniformly in a multi-D volume V.
- Basic theorem of Monte Carlo integration:

$$\int_V f\,dV \simeq V<f> \pm V\sqrt{\frac{<f^2> - <f>^2}{N}},$$

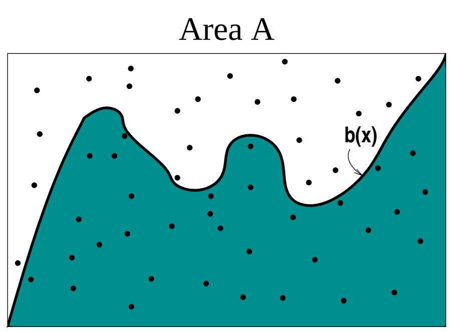
  where

$$<f> \equiv \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i) \quad \text{and} \quad <f^2> \equiv \frac{1}{N}\sum_{i=1}^{N} f^2(\mathbf{x}_i).$$

- The $\pm$ term is a 1-σ error estimate, not a rigorous bound.
- Previous formula works fine if V is simple.
- What if we want to integrate a function g over a region W that is *not* easy to sample randomly?
- Solution: find a simple volume V that *encloses* W and define a new function $f(\mathbf{x})$, $\mathbf{x} \in V$, such that:

$$f(\mathbf{x}) = \begin{cases} g(\mathbf{x}) & \text{for all } \mathbf{x} \in W, \\ 0 & \text{otherwise.} \end{cases}$$

  E.g., suppose we want to integrate $g(x, y)$ over the shaded area inside area A below:



  To integrate, take random samples over the whole rectangle, set

$$f(x_i, y_i) = \begin{cases} g(x_i, y_i) & y_i \le b(x_i), \\ 0 & \text{otherwise,} \end{cases}$$

  and compute

$$\int_{\text{shaded area}} g(x, y)\,dx\,dy \simeq \frac{A}{N}\sum_i f(x_i, y_i).$$

- Nifty example: π can be estimated by integrating

$$p(x,y) = \begin{cases} 1 & x^2+y^2 \leq 1, \\ 0 & \text{otherwise,} \end{cases}$$

over a 2×2 square:

$$\begin{aligned} \pi &= \int_{-1}^{1}\int_{-1}^{1} p(x,y)\,dx\,dy \\ &\simeq \frac{4}{N}\sum_i p(x_i,y_i). \end{aligned}$$

- See *NRiC* for another worked example.

- Optimization strategy: make V as close as possible to W, since zero values of f will increase the *relative* error estimate.

- Principal disadvantage: accuracy increases only as square root of N.

- Fancier routines exist for faster convergence: *NRiC* §7.7–7.8.

- Monte Carlo techniques used in a variety of other contexts: anywhere statistical sampling is useful. E.g.,
  - Predicting motion of bodies with short Lyapunov times if starting positions and velocities poorly known.
  - Determining model fit significance by testing the model against many sets of random synthetic data with the same mean and variance.

Numerical Integration (Quadrature)

- *NRiC* §4.

- Already seen Monte Carlo integration.

- Can cast problem as a differential equation (DE):

$$I = \int_a^b f(x)\,dx$$

is equivalent to solving for $I \equiv y(b)$ the DE $dy/dx = f(x)$ with the boundary condition (BC) $y(a) = 0$.

  - Will learn about ODE solution methods next class.

Trapezoidal and Simpson's rules

- Have abscissas $x_i = x_0 + ih$, $i = 0, 1, \ldots, N+1$.
- A function $f(x)$ has known values $f(x_i) = f_i$.
- Want to integrate $f(x)$ between endpoints a and b.
- Trapezoidal rule (2-point closed formula):

$$\int_{x_1}^{x_2} f(x)\,dx = h\left[\frac{1}{2}f_1 + \frac{1}{2}f_2\right] + \mathcal{O}(h^3 f''),$$

  i.e., the area of a trapezoid of base h and vertex heights f_1 and f_2.

- Simpson's rule (3-point closed formula):

$$\int_{x_1}^{x_3} f(x)\,dx = h\left[\frac{1}{3}f_1 + \frac{4}{3}f_2 + \frac{1}{3}f_3\right] + \mathcal{O}(h^5 f^{(4)}).$$

Extended trapezoidal rule

- If we apply the trapezoidal rule $N-1$ times and add the results, we get:

$$\int_{x_1}^{x_N} f(x)\,dx = h\left[\frac{1}{2}f_1 + f_2 + f_3 + \ldots + f_{N-1} + \frac{1}{2}f_N\right] + \mathcal{O}\left[\frac{(b-a)^3 f''}{N^2}\right].$$

- Big advantage is it builds on previous work:
  - Coarsest step: average f at endpoints a and b.
  - Next refinement: add value at midpoint to average.
  - Next: add values at $\frac{1}{4}$ and $\frac{3}{4}$ points.
  - And so on. This is implemented as `trapzd()` in *NRiC*.

More sophistication

- Usually don't know N in advance, so iterate to a desired accuracy: `qtrap()`.
- Higher-order method by cleverly adding refinements to cancel error terms: `qsimp()`.
- Generalization to order $2k$ (*Richardson's deferred approach to the limit*): `qromb()`.
  - Uses extrapolation methods to set $h \to 0$.
- For improper integrals, generally need *open formulae* (not evaluated at endpoints).
- For multi-D, use nested 1-D techniques.