Evaluating integrals by the Monte Carlo technique.

Consider an integral of the form
\[ \int_a^b g(x) f(x) \, dx \] (\#)

where \( g(x) \geq 0 \) and is normalized such that \( \int g(x) \, dx = 1 \). 

Then, \( g(x) \) can be interpreted as a probability distribution function (pdf) and the integral (\#) as a weighted average of \( f \) over that pdf.

Thus, if we draw a sequence of random values of \( x \in [a,b] \) which have pdf \( g(x) \), we can approximate the integral as
\[ \int_a^b g(x) f(x) \, dx \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i) \, , \text{ where } x_i \text{ are drawn from } g(x) \]

This is what is meant by the Monte Carlo integration.

Of course, we can always assume that \( g(x) = \text{uniform pdf} \)
\[ g(x) = \frac{1}{b-a} \] , then integral of any function \( f(x) \) is approximated by
\[ \int_a^b f(x) \, dx \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i) \, , \text{ where } x_i \text{ are uniformly distributed over } [a,b] \]

However, this is usually very inefficient as points are distributed in regions where \( f(x) \) is small and do not contribute much to the integral. This is similar to integrating with equally spaced intervals for sharply peaked, or rapidly varying integrands.

Thus, it often pays to be able to sample \( g(x) \).

Sampling pdfs is very useful in many other applications.
Error of MC integration: note that \( \frac{1}{N} \sum_{i=1}^{N} f(x_i) = \langle f \rangle \)

at the same time we are computing \( \langle f \rangle \), we can compute

\[ \langle f^2 \rangle = \frac{1}{N} \sum_{i=1}^{N} f(x_i)^2 , \quad \text{and variance} \quad \sigma^2 = \langle f^2 \rangle - \langle f \rangle^2 \]

which will give the uncertainty of the evaluated integral:

\[ \sigma_{\langle f \rangle} = \frac{\sigma}{\sqrt{N}} \]
Example (maint_ex.py maint_ex.py)

Integrate:

\[
\int_0^1 \int_0^{2\pi} \int_0^{2\pi} (e^{-\frac{W}{kT}} - 1) r^2 \sin \beta \sin \theta \, d\theta \, d\phi \, dx \, dy \, dz
\]

\(k=1, T=1\) \quad \Rightarrow \quad \text{analytical result} = \frac{16 \pi^5}{3}

\(W=-\ln(\Theta)\)

This is an integral of the 2nd virial coefficient that enters into the virial expansion of pressure of a multi-particle system:

\[
\frac{p}{kT} = n + B_2(T) n^2 + B_3(T) n^3 + \ldots
\]

where \(n\) is number density of particles

Here \(B_2(T) = -2\pi \int \left[ \exp \left(-\frac{U(r_1)}{kT} \right) \right] r^2 \, dr_1\)

for 2 particles at \(r=r_1\) and \(r_2=0\)
Generating random numbers with a given pdf

Suppose we want random numbers with exponential pdf \( p(y) \, dy = e^{-y} \, dy \). If we have some other pdf \( q(x) \), where \( x = f(y) \), we must have \( p(y) \, dy = q(x) \, dx \)

\[
p(y) = q(x) \left| \frac{dx}{dy} \right|.
\]

Let's take \( q(x) = 1 \), which is a uniform random pdf. and \( y = -\ln(x) \Rightarrow dy = -\frac{dx}{x} = \frac{dx}{e^y} \)

\[
p(y) = e^{-y}.
\]

I.e. if we generate a sequence of uniform numbers and \( -\ln(x) \) (log them with minus sign, or \( \ln(1/x) \)), \( -\ln(x) \) numbers will be distributed with pdf \( p(y) = e^{-y} \)

This is an illustration of the general principle behind generation of pdfs based on the "fundamentally Sampling theorem": for a pdf \( f(x) \) normalized such that \( \int f(x) \, dx = 1 \),

and its corresponding cumulative distribution (cdf) \( F(x) = \int f(x) \, dx \), which is guaranteed to be monotonically increasing, because \( f(x) > 0 \),

if \( x \) be distributed with pdf \( f(x) \Rightarrow F(x) \) is distributed uniformly.

So, for any \( F(x) \) which can be inverted analytically or numerically into \( x(F) \): generate a uniformly distributed sequence \( 2F_{i+1} \rightarrow x(F_{i+1}) \) will have \( f(x) \) pdf.
The rejection technique for sampling

If we do not know normalization of pdf \( f(x) \) on an interval (for whatever reason), but only know its max. value on that interval, we can use reflection technique:

define \( \tilde{f}(x) = \frac{f(x)}{f_{\text{max}}(a \leq x \leq b)} \)

1. Choose a random \( x \) uniformly distributed in \([a, b]\)
2. Choose a second uniformly distributed number \( y \)
3. If \( \tilde{f}(x) > y \) return the value \( x \) picked in step 1. Otherwise, discard it and go back to step 1.

If we don't know \( f_{\text{max}} \) but can define another function \( g(x) > f(x) \) for \( x \in [a, b] \). Rejection technique can be used as follows (if we know cdf of \( g(x) \))

1. Generate \( x \) of the pdf \( g(x) \)
2. If \( x \leq f(x) \), accept it
   \( \Rightarrow \) if \( x > f(x) \), reject it
3. Go to step 1.

The ratio of accepted to rejected curves will be equal to the ratio of areas under curves \( g(x) \) and \( f(x) \)
MCMC sampling

In multi-dimensional problems usual sampling techniques may often be difficult to apply. Also, we often do not know much about the function describing the pdf, and we need to reconstruct it as we sample it. For example, in parameter estimation from data, Bayesian approach is often used to estimate the posterior distribution for model parameters given some data \(d\):

\[
\pi(\vec{x}) = p(d|\vec{x}) p(\vec{x}),
\]

here \(p(\vec{x})\) is the prior distribution assumed for values of \(\vec{x}\) based on some previous info or insights, and \(p(d|\vec{x})\) is likelihood (probability) of "observing" or measuring a set of data \(d\) for values \(\vec{x}\) of the model parameters.

MCMC = Markov Chain Monte Carlo, samples \(\pi(\vec{x})\) by generating a Markovian chain of \(\vec{x}\) values, in which step \(\vec{x}\) depends only on \(\vec{x}_{i-1}\), and not any prior points in the chain (which is the definition of Markovian random process).

The key ingredients are:

- how to choose a step from \(\vec{x}_{i-1}\) to \(\vec{x}_i\)
- criteria on whether to accept or reject a given step
Metropolis-Hastings algorithm (see Ch. 15 of NR)

Developed in the early 1950s, to simulate thermodynamic systems of many particles.

- Choose random steps \( \delta \) in all components of \( \mathbf{x} \) at the same time uniformly distributed in \(-\delta_i; \delta_i\).

- Accept step with probability \( p(\mathbf{x}_c | \mathbf{x}_{i-1}) \) that is designed to satisfy equilibrium condition:

\[
\frac{\pi(\mathbf{x}_{i-1}) p(\mathbf{x}_c | \mathbf{x}_{i-1})}{\pi(\mathbf{x}_i) p(\mathbf{x}_i | \mathbf{x}_{i-1})} = \frac{\pi(\mathbf{x}_{i-1}) q(\mathbf{x}_i | \mathbf{x}_c)}{\pi(\mathbf{x}_c) q(\mathbf{x}_c | \mathbf{x}_{i-1})}
\]

Specifically, generate a step starting with \( \mathbf{x}_i \) to \( \mathbf{x}_c \), calculate acceptance probability:

\[
\alpha(\mathbf{x}_i, \mathbf{x}_c) = \min \left[ 1, \frac{\pi(\mathbf{x}_c) q(\mathbf{x}_i | \mathbf{x}_c)}{\pi(\mathbf{x}_i) q(\mathbf{x}_c | \mathbf{x}_i)} \right]
\]

\( \rightarrow \) accept \( \mathbf{x}_c \) with probability \( \alpha \) — i.e. draw a uniform random number \( r \in (0, 1) \), if \( r < \alpha \), accept \( \mathbf{x}_c \), if \( r \leq \alpha \), draw a new \( \mathbf{x}_c \) to try.

\( q \) here is the proposal pdf, in the simplest case can be taken \( q = 1 \)