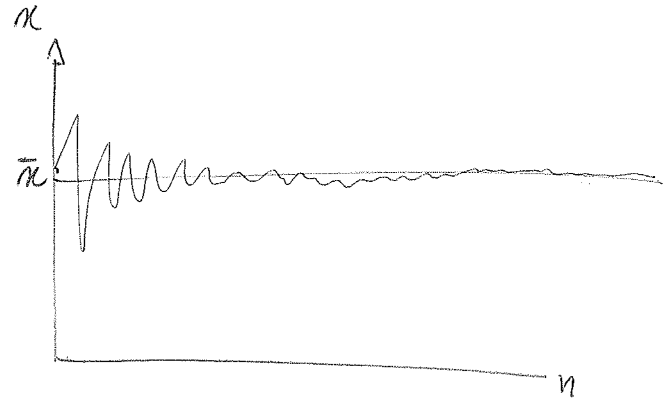
Monte Carlo Methods

- Convergence in probability

The sequence $\{x_1, \dots, x_n\}$ is said to converge in probability to $\bar{x}$ if, $\forall \varepsilon > 0$ and $\forall \eta > 0$, a value n_0 can be found such that:

$$\boxed{P(|x_n - \bar{x}| > \varepsilon) < \eta \quad \forall n \geq n_0}$$



- Law of large numbers

Assume to repeat the same measurement n times, where outcome is a random variable x with a given PDF and STD σ.

The average will be: $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$

→ Weak law: If the mean μ exists, ~~m~~ and if $\lim_{n\to\infty} \left[\frac{1}{n^2} \sum \sigma_i^2 \right] = 0$

Then $\bar{x}$ converges to μ in quadratic mean: $\lim_{n\to\infty} E\left[(\bar{x} - \mu)^2 \right] = 0$

→ Strong law: If $\lim_{n\to\infty} \left[\sum_i \left(\frac{\sigma_i}{i} \right)^2 \right]$ is finite

Then $\bar{x}$ converges almost certainly to μ,

which means that $P\left(\lim_{n\to\infty} \bar{x} = \mu \right) = 1$

Take home message: if the parent mean μ exists, the more you measure, the the closer the sample mean $\bar{x}$ will go to μ.

- Central limit Theorem

Recall that if we have a sequence of independent variables x_i, each with a distribution with mean μ_i and variance σ_i^2, the distribution of the sum ~~S = Σx~~ $S = \sum x_i$ will have mean $\mu = \sum \mu_i$ and $\sigma^2 = \sum \sigma_i^2$.

The central limit Theorem states that:

$$\boxed{\lim_{n \to \infty} \frac{S - \sum_{i=1}^{n} \mu_i}{\sqrt{\sum_{i=1}^{n} \sigma_i^2}} = \text{Gaus}(0, 1)}$$

In other words, ~~the~~ sum of ~~any~~ n random variables tend to a Gaussian.
Or, as Shikong said, "the end of the world is Gaussian"

<u>Example: Gaussian random number generator</u>

- Pseudorandom numbers and Monte-Carlo methods

So far, we have used random numbers in almost all exercises, without actually explaining what they are and what are their properties and limitations.

A pseudorandom number generator is an algorithm that generates a sequence of numbers distributed according to some PDF and that resemble very closely an actual distribution of random numbers with the same PDF.

Properties of pseudorandom generators:

- each extraction must be statistically independent from the previous ones.
- all extractions should be distributed according to the same PDF:

$$f(x_i) = f(x_j) \quad \forall\, i, j$$

$$f(x_i | x_{i-m}) = f(x_i) \quad \forall\, i, m$$

- after a given period p, the sequence will repeat itself: $x_{i+p} = x_i$.
  Obviously, we want p to be as large as possible.
  In this sense, the distribution of n pseudo random numbers can be considered to be truly random up to $n = p$.

→ We should be able to initialize the generator in such a way that it can reproduce exactly the same sequence of random numbers.
The inizialization is commonly done by passing a user-defined number called seed.
This is very useful for debugging code.

- Uniform random number generators

Most common ~~generator~~ and simple generator produces numbers in $[0;1[$

For example, the lrand48 (standard of C) uses the following algorithm:

$$x_{i+1} = (a x_i + c) \bmod (m)$$

where: $m = 2^{48}$

$a = 25214903917$

$c = 11$

The produced random numbers are uniformly distributed between 0 and $2^{48}-1$, and mapped into floating-point ~~nub~~ numbers between 0 and 1.

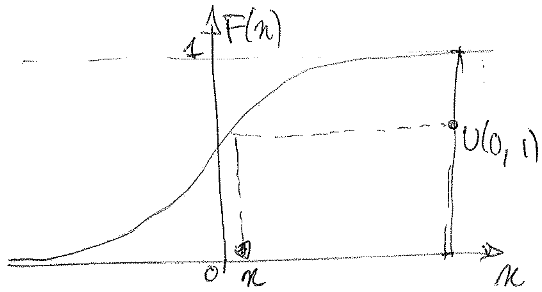
→ A uniform random number can be remapped to any other interval $[a, b[$ simply by doing:

$$x \to x' = a + x(b-a)$$

- Non-uniform generators: inverse-transformation method

Suppose we want to generate a random number distributed as $f(x)$.

The cumulative $F(x)$ will map $x \to [0, 1[$



Therefore we can invert (analitically or numerically) $F(x)$ to obtain a number distributed as $f(x)$.

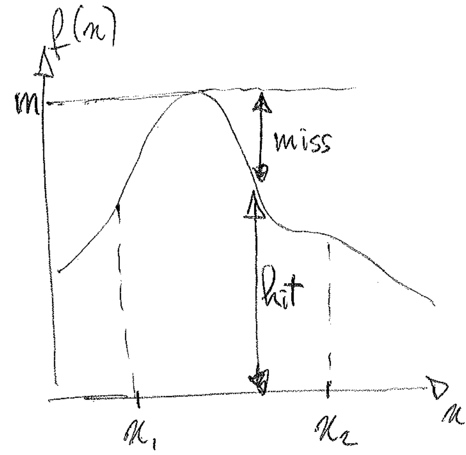
1) Generate r uniformly in $[0, 1[$
2) Invert F: $x = F^{-1}(r)$

• Gaussian generator using central-limit theorem

→ It works, but 1) it's fucking inefficient
2) it's truncated

• Acceptance-rejection method (hit-or-miss MC)

Assume a PDF $f(x)$ defined in interval $[x_1, x_2[$



Assume we know $m = \max\{f(x),\ x \in [x_1, x_2[\}$

Then we can do the following:

1) Generate a uniform number x in $[x_1, x_2[$
2) Generate a uniform number r in $[0, m[$
3) If $f(x) > r$, we accept x, otherwise we go back to 1
4) Populate a histogram with the accepted values. This will approximate the original $f(x)$ for large numbers of generated points.

Properties: → The method works well if we can compute $f(x)$ fast enough, but if $F(x)$ is computationally intensive

→ We only get a binned approximation of $f(x)$

→ The efficiency of the method is:

$$\varepsilon = \frac{\int_{x_1}^{x_2} f(x)\,dx}{(x_2 - x_1)\cdot m}$$

Therefore, it's inefficient if $f(x)$ is very peaked.

→ It can be applied to multi-dimensional cases, but leads to the "curse of dimensionality".

Example: acceptance-rejection method

- Combination of random ~~number generators~~ variables

For complicated cases, we can combine the transform-rejection method and the acceptance rejection method. ~~Or we can us~~

We can also use random number generators to produce binned PDFs of variables that are a combination of other ~~ones~~ variables.
This is for example the case of the ratio of two variables.

<u>Example: Change of variable → ratio</u>

- Numerical integration with MC

The acceptance-rejection method estimates the integral $\int_{x_1}^{x_2} f(x)\,dx$ from the fraction of accepted events k over the number n of generated events:

$$I = \int_{x_1}^{x_2} f(x)\,dx \simeq (x_2 - x_1) \cdot \frac{k}{n}$$

~~This applies also for multi-dimensional integration.~~

The uncertainty is (we'll see it in some future lecture):

$$\sigma_{\hat{I}} = (x_2 - x_1)\sqrt{\frac{\hat{I}(1-\hat{I})}{N}}$$

Notice that there is no dependence on the dimensionality.
This makes MC method ~~(relati~~ advantageous ~~fo~~ when computing the integrals of high-dimensionality PDFs.

The problem, however, ~~it~~ might be to find the maximum of $f(x)$.

- Markov Chain Monte Carlo

Some probability distributions can be sampled more efficiently by producing sequences of correlated pseudorandom numbers, where each x_i depends on the previous m extractions.

A sequence of random variables $x_0, \dots, x_n$ is a Markov chain if the PDF obeys to:

$$f(x_{n+1}; x_0, \dots, x_n) = f(x_{n+1}; x_n)$$

i.e. if $f(x_{n+1})$ depends only on the immediately previous extraction.
This works in any dimension.

- Metropolis - Hastings

A common MCMC algorithm is Metropolis-Hastings.
Suppose we want to sample a PDF $f(\vec{x})$.

We do the following:

1) Pick a point $\vec{x}_0$ ~~randomly~~ uniformly distributed in the sample space Ω.
2) Evaluate $f(\vec{x}_0)$
3) Generate a second point $\vec{x}$ according to a predefined PDF $q(\vec{x}, \vec{x}_0)$, called "proposal distribution".
4) Evaluate $f(\vec{x})$
5) Generate a uniform number $u \in [0, 1[$
6) If $$\frac{f(\vec{x})\, q(\vec{x}_0, \vec{x})}{f(\vec{x}_0)\, q(\vec{x}, x_0)} > u$$ accept the points and set $\vec{x}_1 = \vec{x}$

   Otherwise reject $\vec{x}$
7) Iterate back to (3), substituting $\vec{x}_0 \to \vec{x}_1$ if the point was accepted.

Notice that: if $q(\vec{x}, \vec{x}_0) = q(\vec{x}_0, \vec{x})$, the condition (6) can be simplified.
Usually the proposal function is a multivariate with a fixed STD.

Properties:

→ Metropolis-Hastings allows to map an n-dimensional PDF.
The condition (6) ensures ~~that~~ that if we move to a higher point, the move is always accepted, but ~~also move~~ at the same time we have a small but non-zero probability to accept also lower points.

→ Metropolis-Hastings does not ~~go~~ always find the mode of the distribution!
It will find it if you ~~are in lo~~ have few dimensions, but it won't if you have dim $\gtrsim 10$ (by experience).

Examples: → Drunk Markov in golf course
→ Random number correlation