Monte Carlo methods and event generators

- **Basic of MC simulation**
- **Event generators**

Following:
- **Geant4 tutorials on MC basic**
- **T. Sjostrand lectures on MC evetn generators**

Prof. dr hab. Elżbieta Richter-Wąs

Monte Carlo method applications

Monte Carlo applications:

- **Physics**: particle physics, astrophysics, nuclear physics, radiation damage,...
- **Medicine**: radiation therapy, nuclear medicine, computer tomography,…
- **Chemistry**: molecular modeling, semiconductor devices,...
- **Finance**: financial market simulations, pricing, forecast sales, currency,…
- **Optimization problems**: manufacturing, transportation, health care, agriculture,...
- **Data production** for **neural nets**
- And **much more**!

MC vs **Neural Nets**: slower but more precise and controllable

The simplest MC example: probabilities of rulette

What is the probability of red?

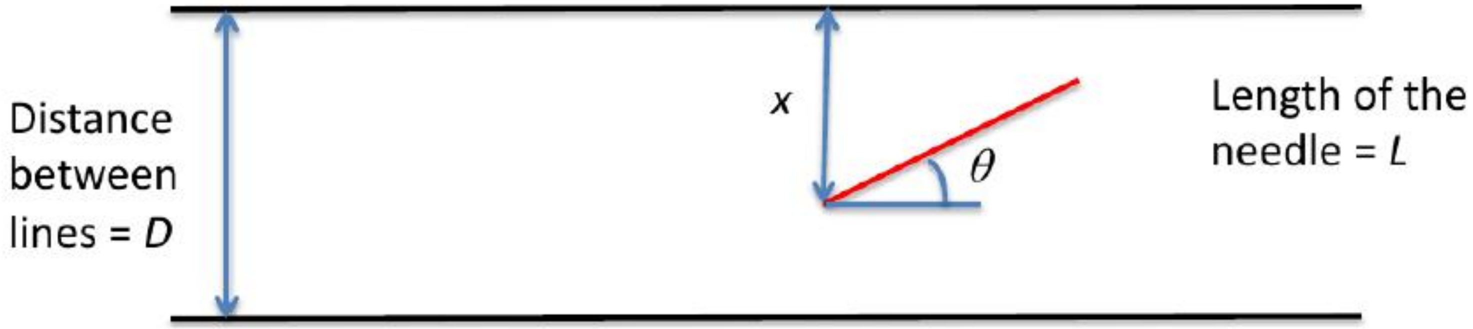
- Observe the result **many times** (it is not necessary to stake:)
- Count the total of red wins: N_{red}
- Count the total of games: N_{total}
- The measured probability of red will be: $P_{red} = N_{red}/N_{total}$
- If $N_{total} \rightarrow \infty \Rightarrow P_{red} \rightarrow P_{red\ true} = 18/(18+18+1) = 0.486$

MC example: Buffon's Needle (1977)

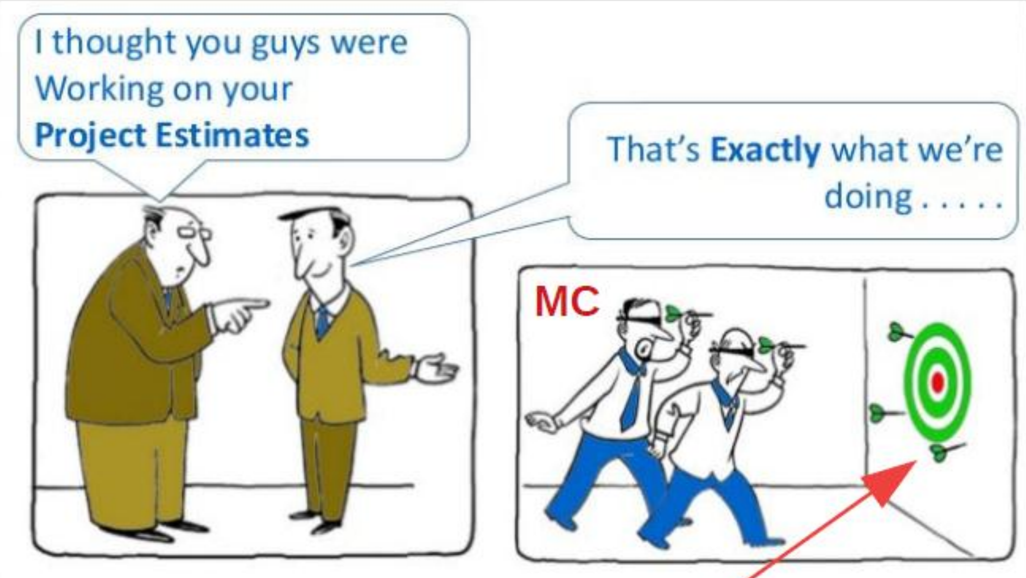
- One of the oldest problems in the field of geometrical probability, first stated in 1777.
- Drop a needle on a lined sheet of paper and determine the probability of the needle crossing one of the lines
- Remarkable result: probability is directly related to the value of π
- The needle will cross the line if $x \leq L \sin(\vartheta)$. Assuming $L \leq D$, how often will this occur?

$$P_{cut} = \int_0^{\pi} P_{cut}(\theta) \frac{d\theta}{\pi} = \int_0^{\pi} \frac{L \sin\theta}{D} \frac{d\theta}{\pi} = \frac{L}{\pi D} \int_0^{\pi} \sin\theta \, d\theta = \frac{2L}{\pi D}$$

- By sampling P_{cut} one can estimate π.



MC is a simple and a general method



The **Monte Carlo (MC)** method is a method to obtain **deterministic results** from **random** values

In other words, **try many times** and **count** the **total** of the outcomes you like

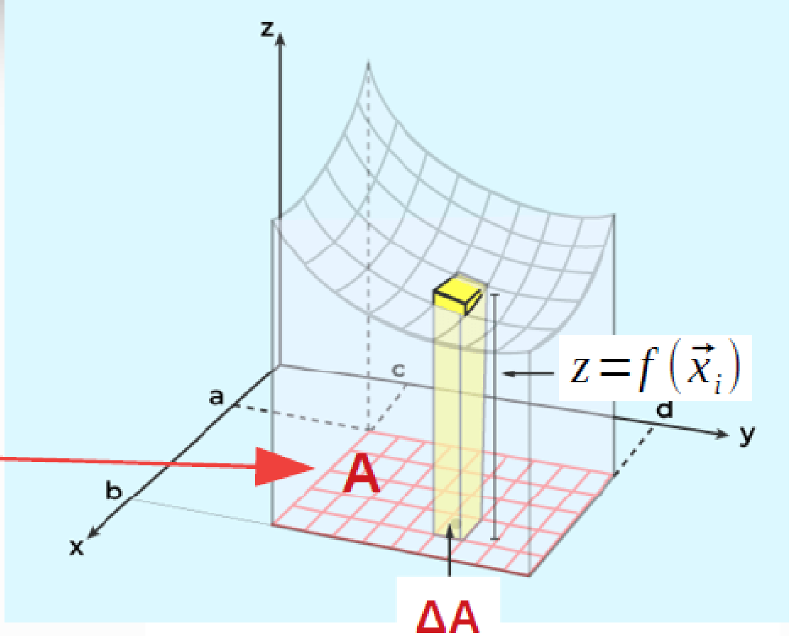
- Generate **N random points** $\vec{x}_i$ in the problem space
- Calculate the **score** $f_i = f(\vec{x}_i)$ for the N points
- Calculate the **result** of your **average score**: $\bar{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$
- According to the **Central Limit Theorem**, $\bar{f}$ will approach the **true** average value $\langle f \rangle = \lim_{N \to \infty} \bar{f}$

Monte Carlo numerical integration: extremely useful for multidimensional integrals!

$$A = \int_A d\vec{x}_i; \quad d\vec{x}_i = dx_{1i} dx_{2i} dx_{3i} \ldots = dA$$

$$I = \int_A f(\vec{x}_i) d\vec{x}_i - ?$$

Idea is exactly the same!



$$\Delta A = \frac{A}{N}$$

- Generate **N random points** in $\vec{x}_i \in A$
- Calculate the **score** $f_i = f(\vec{x}_i)$ for the N points
- Calculate the **result** of your **integral**:

$$I = \int_A f(\vec{x}_i) d\vec{x}_i \approx I_{MC} = \sum_{i=1}^{N} f_i \Delta A = \frac{A}{N} \sum_{i=1}^{N} f_i = A \bar{f}$$

- Following the **Central Limit Theorem**, I_{MC} will approach the **true** integral value:

$$I = \int_A f(\vec{x}_i) d\vec{x}_i = \lim_{N \to \infty} I_{MC} = A \lim_{N \to \infty} \bar{f}$$

MC example: Laplace's method of calculting π (1886)

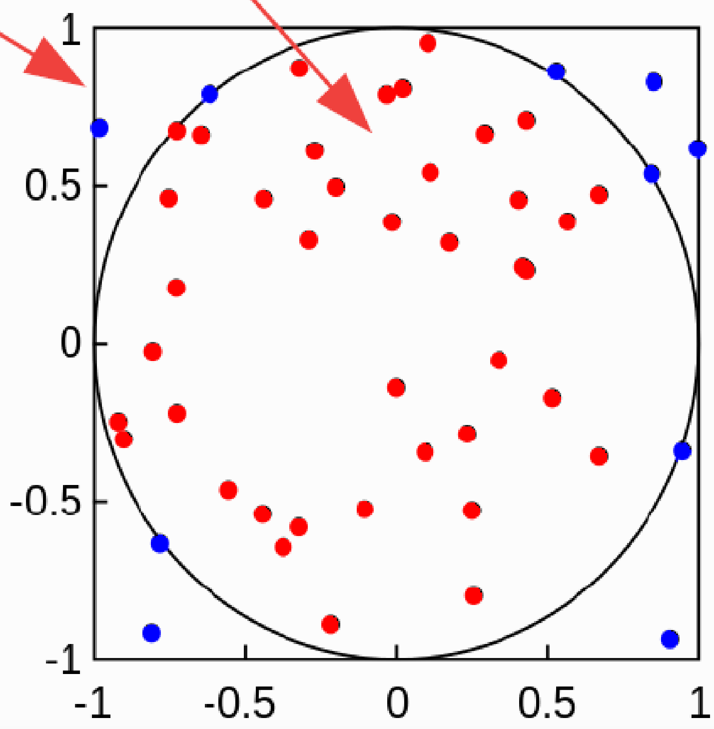
- **Side** of the square = 1
- Area of the **square** = **A** = 4
- Area of the **circle** is integral we are calculating: $I=\pi$

$$f_i=f(\vec{x}_i)=\begin{cases}1, if\ \vec{x}_i\in I\\0, if\ \vec{x}_i\notin I\end{cases}$$

- Everything we need is to **count** the number of points $\vec{x}_i$ inside the circle: $N_c=N_{\vec{x}_i\in I}=\sum_{i=1}^{N} f_i$

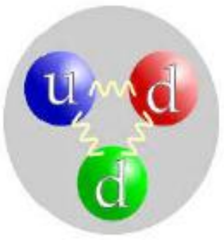
- This will give the value of our integral:

$$I_{MC}=\frac{A}{N}\sum_{i=1}^{N} f_i=\frac{4}{N}N_c\underset{N\to\infty}{\to}\pi$$



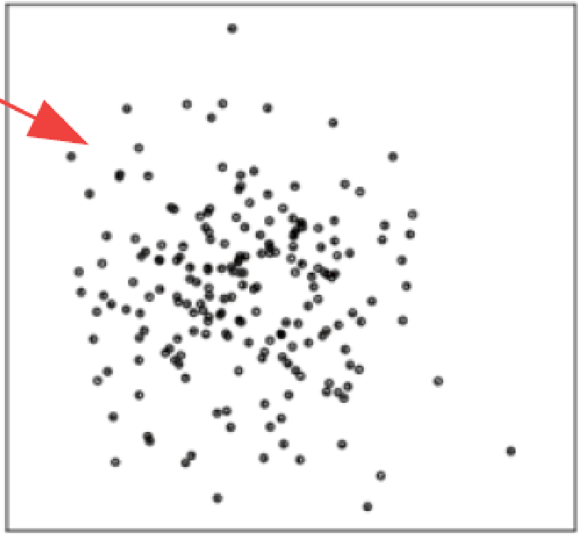
History of MC methods

- Fermi (1930): random method to calculate the properties of the newly discovered neutron
- Manhattan project (40's): simulations during the initial development of thermonuclear weapons. Von Neumann and Ulam coined the term "**Monte Carlo**"
- Metropolis (1948) first actual Monte Carlo calculations using a computer (ENIAC)
- Berger (1963): first complete coupled electron-photon transport code that became known as ETRAN
- Exponential growth since the 1980's with the availability of digital computers



Probability Density Function (PDF)

- If we generate a set of **random variables** $\vec{x}_i \in A$, the **probability** of them is **not necessarily equal**. In some zones of *A* we can find more random variables and some of them less.



- However, we can define a function related to the probability of the generated points, so called ***probability density function*** **(PDF)**.

- **Probability Density Function (PDF)** $p(\vec{x}_i)$ of vector $\vec{x}_i$ is a function that has three properties:

For simplicity let's switch to the **1D case**:

1) belongs to some region A: $\vec{x}_i \in A$ — $a \le x \le b$

2) is non-negative in this region: $p(\vec{x}_i) \underset{\vec{x}_i \in A}{\ge} 0$ — $p(x) \underset{a \le x \le b}{\ge} 0$

3) is normalized: $\int_A p(\vec{x}_i) d\vec{x}_i = 1$ — $\int_a^b p(x) dx = 1$

Cumulative Distribution Function (CDF)

PDF IS NOT A PROBABILITY
It is a probability density

Probability is the integral of PDF:

$$Prob\{x_1 \le x \le x_2\} = \int_{x_1}^{x_2} p(x)\,dx$$

- **Cumulative Density Function (CDF)** is a direct measure of probability:

$$F(x) = Prob\{a \le x \le x'\} = \int_{a}^{x} p(x')\,dx'$$

$$Prob\{x_1 \le x \le x_2\} = F(x_2) - F(x_1)$$

- **CDF** has the following **properties**:

1) F(a) = 0, F(b) = 1;

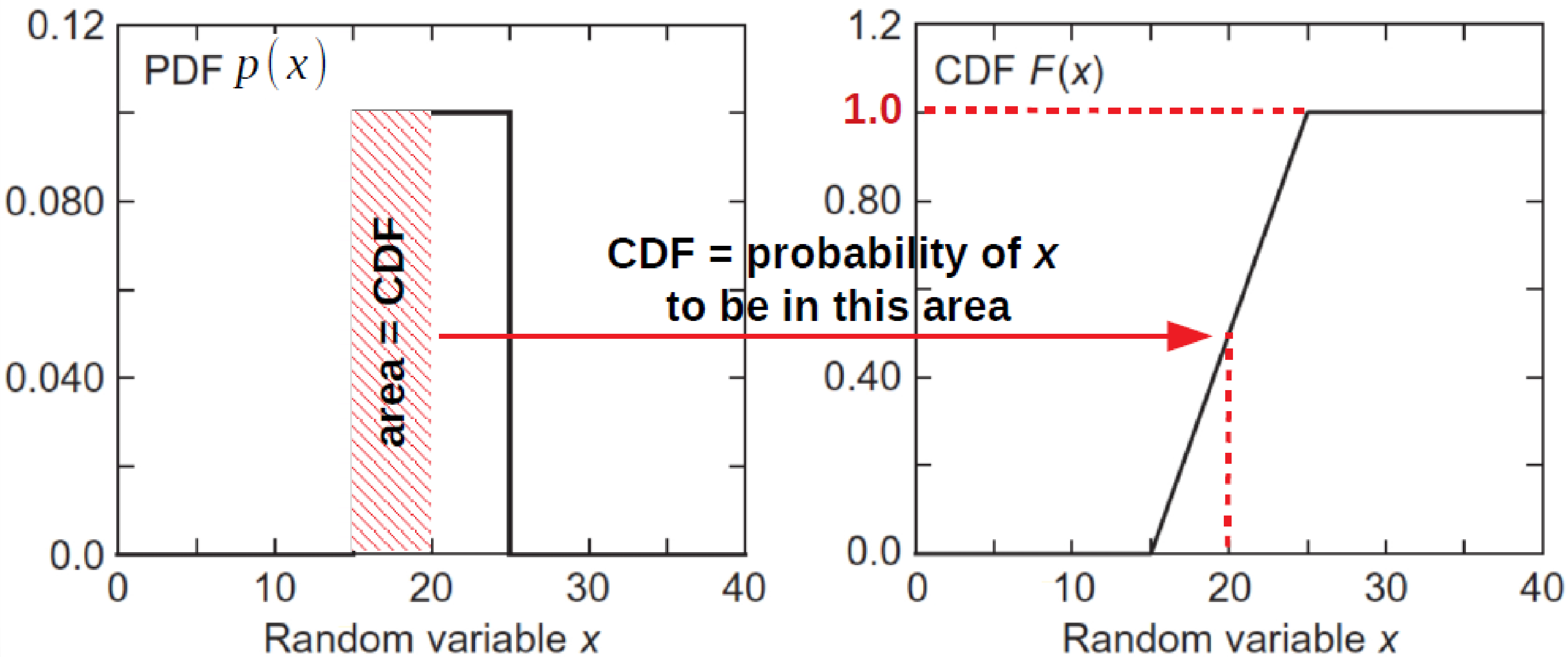
2) F(x) is monotonically increasing, since p(x)≥0.

Some example distribution – Uniform PDF

- The uniform (rectangular) PDF on the interval [a, b] and its CDF are given by

$$p(x) = \frac{1}{b-a}$$

$$F(x) = \int_a^x \frac{1}{b-a} dx' = \frac{x-a}{b-a}$$



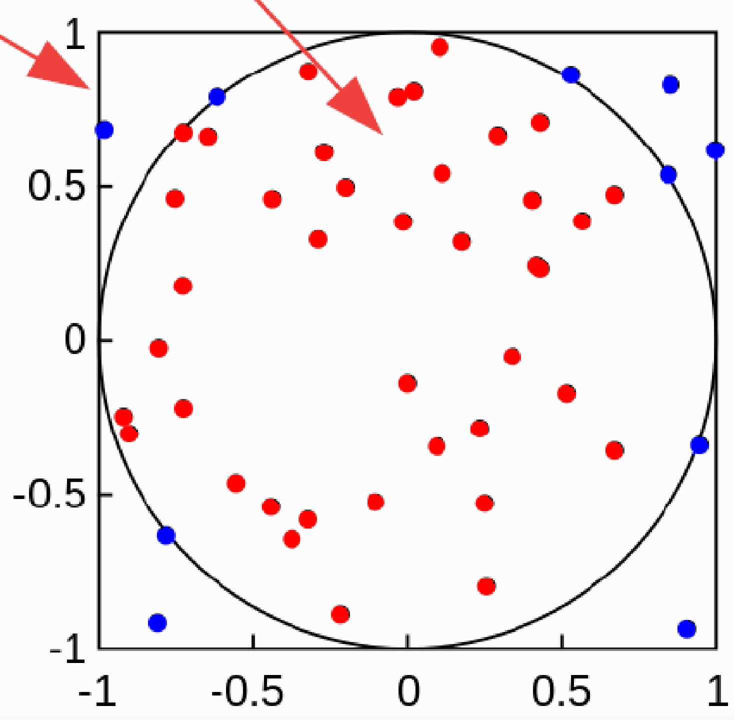
Where we use uniform distribution

- **Side** of the square = 1
- Area of the **square** = **A** = 4
- Area of the **circle** is integral we are calculating: $I = \pi$

$$f_i = f(\vec{x}_i) = \begin{cases} 1, if\ \vec{x}_i \in I \\ 0, if\ \vec{x}_i \notin I \end{cases}$$

- Everything we need is to **count** the number of points $\vec{x}_i$ inside the circle: $N_c = N_{\vec{x}_i \in I} = \sum_{i=1}^{N} f_i$
- This will give the value of our integral:

$$I_{MC} = \frac{A}{N} \sum_{i=1}^{N} f_i = \frac{4}{N} N_c \underset{N \to \infty}{\to} \pi$$

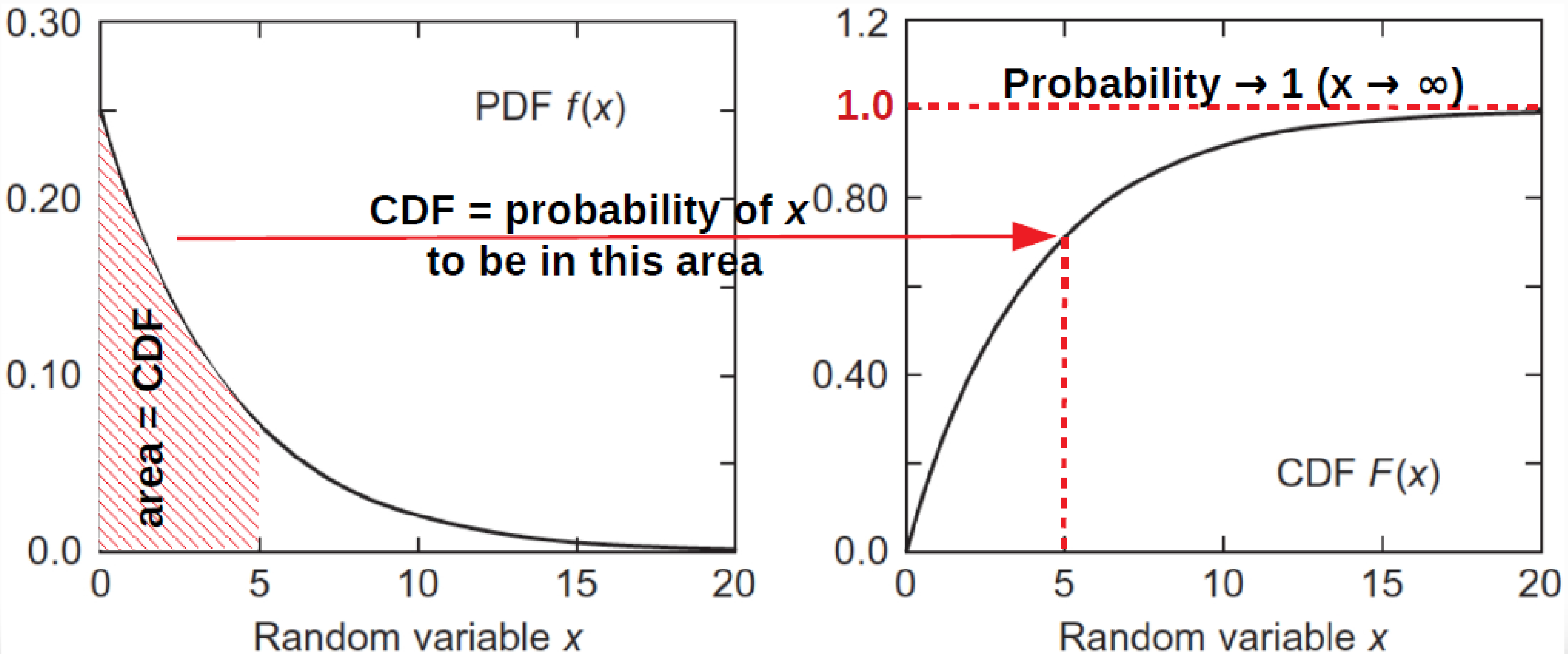


Some example distributions – exponential PDF

- The exponential PDF on the interval [0, ∞] and its CDF are given by

$$p(x) = p(x|a) = \alpha e^{-\alpha x}$$

$$F(x) = \int_0^x \alpha e^{-\alpha x'} dx' = 1 - e^{-\alpha x}$$



Exponential distribution example: nuclear decay

- The time of nuclear decay is a random value with probability density function

$$p(t) = \frac{1}{\tau} e^{-\frac{t}{\tau}}$$

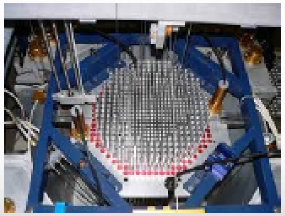
where **τ** is the **mean lifetime** of the nucleus; the **half-life** time **$t_{1/2}$ = τ ln(2)**

- The **probability** of **decay** at time **t** is calculated using the **CDF**:

$$P_{decay}(t) = F(t) = \int_0^t \frac{1}{\tau} e^{-\frac{t'}{\tau}} dt' = 1 - e^{-\frac{t}{\tau}} \in [0,1]$$

- To use Monte Carlo to generate the decay time t one needs to replace $P_{decay}(t)$ by a random number $\xi \in [0,1]$:

$$t = -\tau \ln(1-\xi) = -\tau \ln \xi$$



- **Nuclear decay applications**: nuclear physics, nuclear reactors, nuclear medicine, SPECT, PET, ...

Mean, variance and standard deviation

- Consider a function **z(x)**, where x is a random variable described by a PDF p(x).
- The function **z(x)** itself is a **random** variable. Thus, the **mean** value of z(x) is defined as:

$$\langle z \rangle \equiv \mu(z) \equiv \int_a^b z(x)\, p(x)\, dx$$

- Then, variance of z(x) is given as this

$$\sigma^2(z) = \langle (z(x) - \langle z \rangle)^2 \rangle = \int_a^b (z(x) - \langle z \rangle)^2 p(x)\, dx = \langle z^2 \rangle - \langle z \rangle^2$$

- The heart of a Monte Carlo analysis is to obtain an estimate of a mean value (a.k.a. **expected value**). If one forms the estimate

$$\bar{z} = \frac{1}{N} \sum_{i=1}^{N} z_i = \frac{1}{N} \sum_{i=1}^{N} z(x_i)$$

$$\langle z \rangle = \lim_{N \to \infty} \bar{z}$$

- The variance of $\bar{z}$ is given as

$$\sigma^2(\bar{z}) = \sigma^2\left(\frac{1}{N} \sum_{i=1}^{N} z_i\right) = \frac{1}{N^2} \sum_{i=1}^{N} \sigma^2(z) = \frac{1}{N} \sigma^2(z)$$

Monte Carlo error

- The Monte Carlo error is given by the standard deviation of the expected value:

$$\sigma(\bar{z}) = \frac{\sigma(z)}{\sqrt{N}}; \sigma(z) = \sqrt{\sum_{i=1}^{N} (z_i - \langle z \rangle)^2 / N}$$

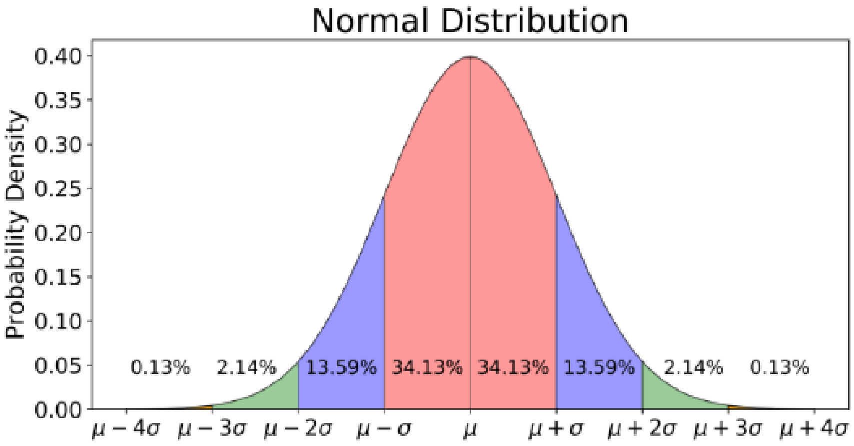
- Since in MC we don't know the true value $\langle z \rangle$, we should use corrected ("unbiased") sample standard deviation:

$$s(z) = \sqrt{\sum_{i=1}^{N} (z_i - \bar{z})^2 / (N-1)}$$

- **Confidence coefficient**:

$$Prob\left\{\bar{z} - \lambda \frac{s(z)}{\sqrt{N}} < \langle z \rangle < \bar{z} + \lambda \frac{s(z)}{\sqrt{N}}\right\} \simeq \frac{1}{\sqrt{2\pi}} \int_{\lambda}^{\lambda} e^{-u^2/2} du$$

λ	confidence coefficient	confidence level
0.25	0.1974	20%
0.50	0.3829	38%
1.00	0.6827	68%
1.50	0.8664	87%
2.00	0.9545	95%
3.00	0.9973	99%
4.00	0.9999	99.99%



Higgs boson **discovery**:
λ=5 («5σ»)

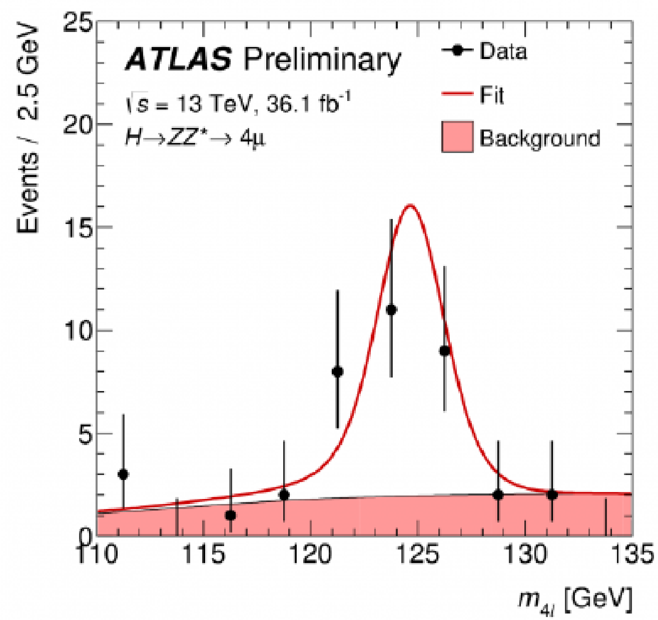
Sometimes statistics is a problem

- **Decay** of an unstable particle itself is a **random process**
- This decay may happen through **different channels** => **Branching ratio:**

$\pi^+ \rightarrow \mu^+ \nu_\mu$	(99.9877 %)
$\pi^+ \rightarrow \mu^+ \nu_\mu \gamma$	(2.00×10^{-4} %)
$\pi^+ \rightarrow e^+ \nu_e$	(1.23×10^{-4} %)
$\pi^+ \rightarrow e^+ \nu_e \gamma$	(7.39×10^{-7} %)
$\pi^+ \rightarrow e^+ \nu_e \pi^0$	(1.036×10^{-8} %)
$\pi^+ \rightarrow e^+ \nu_e e^+ e^-$	(3.2×10^{-9} %)

Very low probability

Higgs boson events* **errorbars**



- The **statistical error** of decay events in a **decay channel** or of the **errorbars** in any **histogram** can be estimated using the same formula:

$$Error(1\sigma) = \sqrt{\frac{p(1-p)}{N}}$$

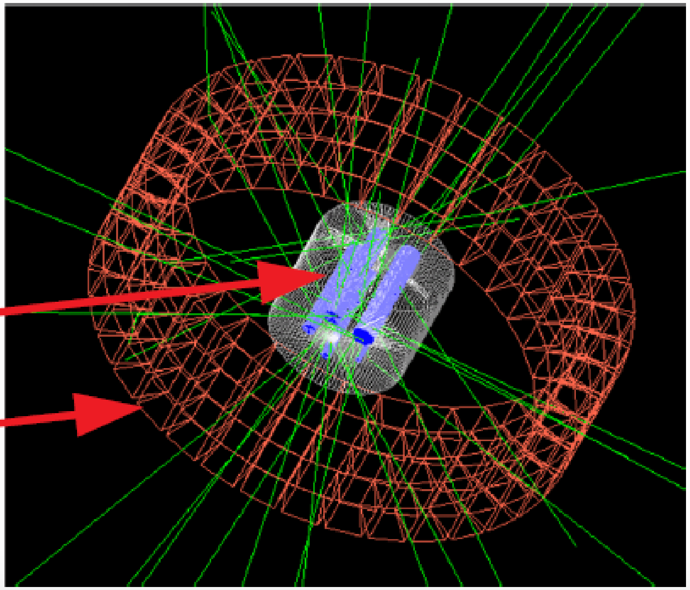
for **3σ** multiply it by 3, confidence level **99%**

Geant4*: a Monte Carlo simulation toolkit

- **Geant4** generates **primary beam** of particles randomly according the distribution set up.
- All the **Geant4** primary particles are simulated independently.
- Primary particles are **tracked** in the material, can **decay** and **produce secondary particles**, for instance **radiation.** This is simulated using various **Geant4 processes** most of which are **random**, which is also illustration of Monte Carlo.
- The **Geant4 output** is some **distribution** of particles as well as **scoring** of interesting events.

In **Positron Emission Tomography** (PET) we have (picture from **):

- a **source** of **gamma**-rays distributed in some space **randomly emitting** the photons and surrounded by some material
- A **detector** to **score** these **gamma**-rays



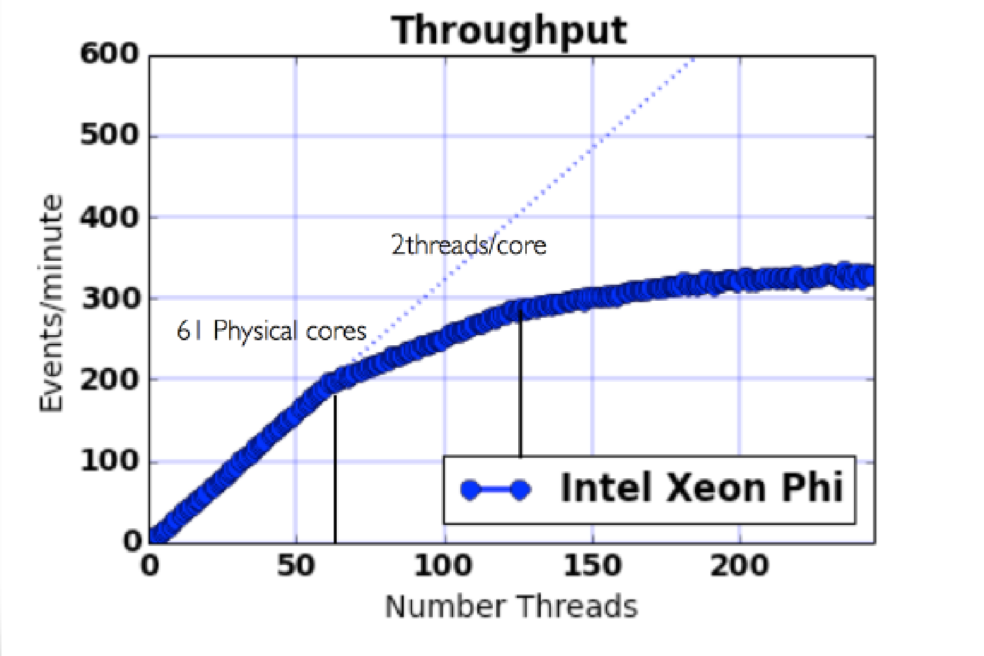
*https://geant4.web.cern.ch/

**D. P. Watts et al. Nature Communications, 12, 2646 (2021)

Monte Carlo parallelization => supercomputing

- All **Monte Carlo** points are **independent** => simple parallelization
- In **Geant4** all primary particles are automatically distributed between different cores of the CPU using **multithreading**
- **Geant4** includes also **MPI parallelization** to parallelize across on **multiple nodes**

Linear scaling on physical cores*



NURION@KISTI (Korea)

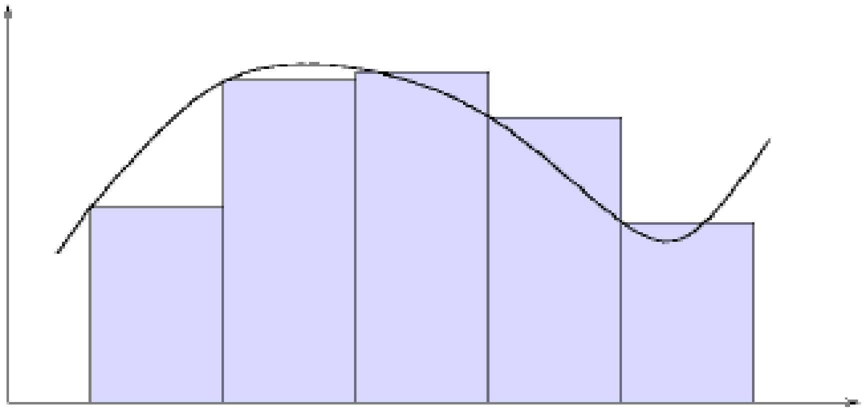
Monte Carlo methods

Monte-Carlo methods generally follow the following steps:

1. Determine the statistical properties of possible inputs
2. Generate many sets of possible inputs which follows the above properties
3. Perform a deterministic calculation with these sets
4. Analyze statistically the results

The error on the results typically decreases as $1/\sqrt{N}$

Numerical integration



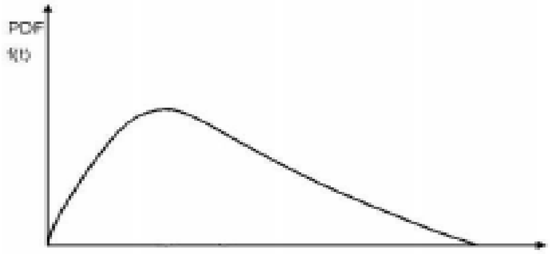
Most problems can be solved by integration

Monte-Carlo integration is the most common application of Monte-Carlo methods

Basic idea: Do not use a fixed grid, but random points, because:

1. Curse of dimensionality: a fixed grid in D dimensions requires N^D points
2. The step size must be chosen first

Error estimation

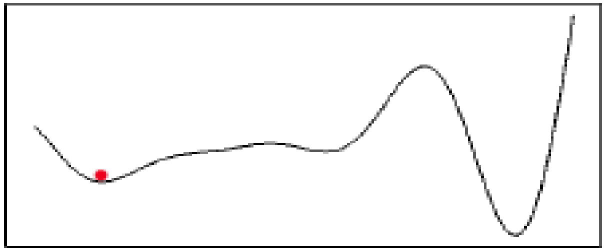


Given any arbitrary probability distribution and provided one is able to sample properly the distribution with a random variable (i.e., $x \sim f(x)$), Monte-Carlo simulations can be used to:

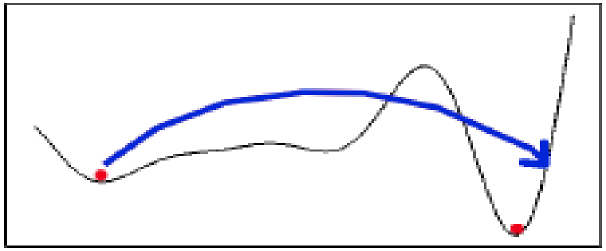
- determine the distribution properties (mean, variance,...)
- determine confidence intervals, i.e. $P(x > \alpha) = \int_{\alpha}^{\infty} f(x)\mathrm{d}x$
- determine composition of distributions, i.e. given $P(x)$, find $P(h(x))$, $h(x) = x^2; cos(x) - sin(x); \ldots$

Note that these are all integrals!

Optimisation problems

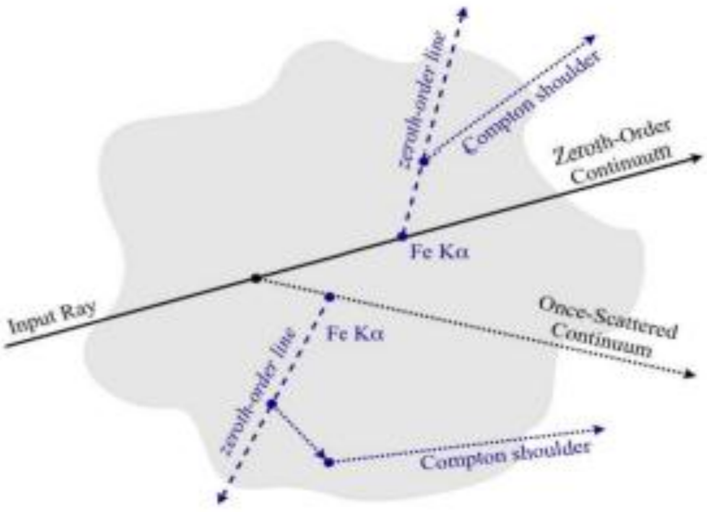


Numerical solutions to optimization problems incur the risk of getting stuck in local minima.



Monte-Carlo approach can alleviate the problem by permitting random exit from the local minimum and find another, hopefully better minimum

Numerical simulations

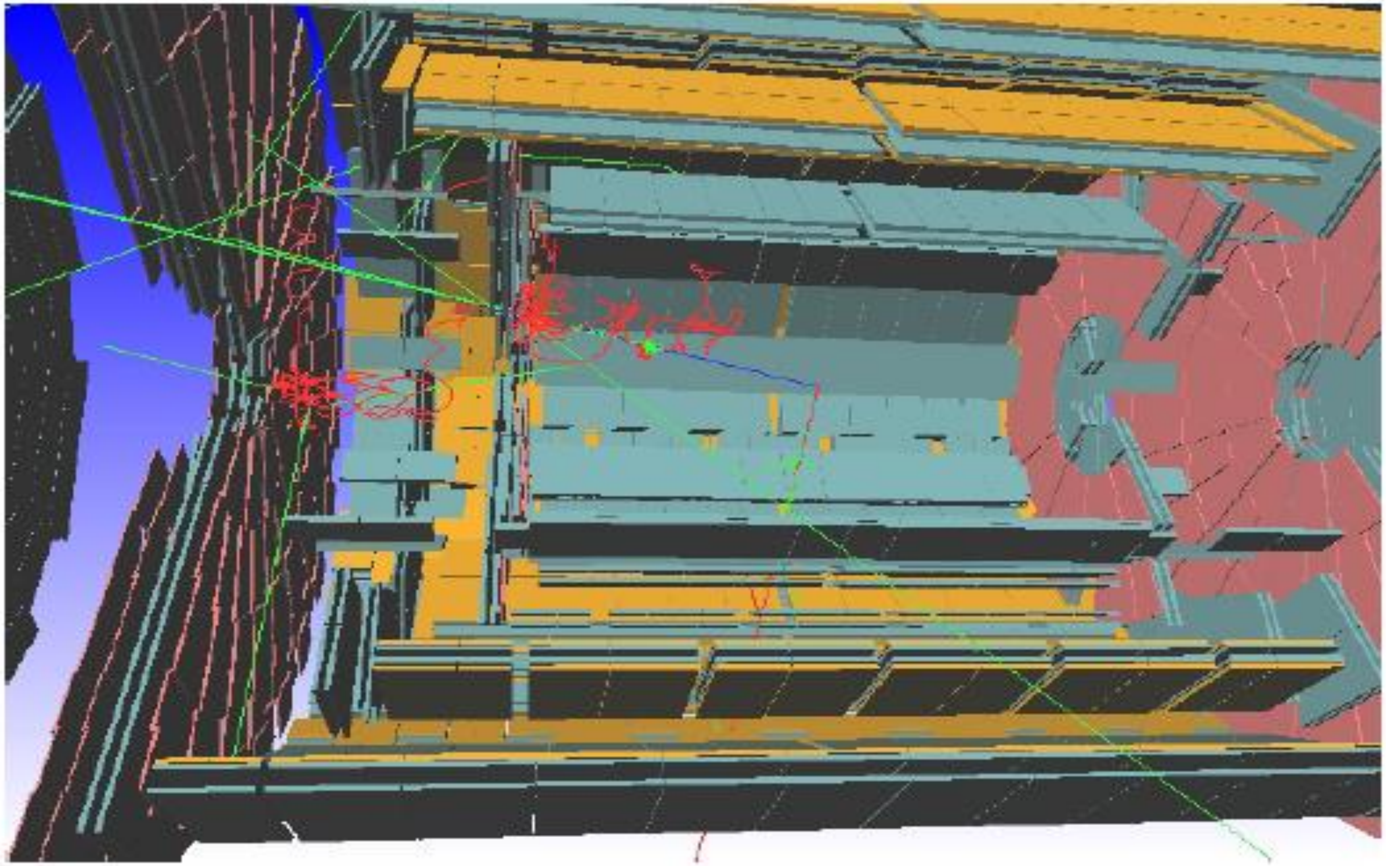


- Radiation transfer is Google-wise the main astrophysical application of Monte-Carlo simulations in astrophysics
- In particle physics and high-energy astrophysics, many more physical processes can be simulated

Some physical processes are discretized and random by nature, so Monte-Carlo is particularly adapted

Numerical simulations

GEANT4



GEANT4 is also used to determine the performance of X-ray and gamma-ray detectors for astrophysics

Random numer generators

Basic principles

- We want to draw many random variables $N_i \sim \mathcal{U}(0,1)$, $i = 1, \ldots$ which satisfy (or approximate sufficiently well) all randomness properties
- $N_i \sim \mathcal{U}(0,1)$, $\forall i$ is not sufficient. We also want that $f(N_i, N_j, \ldots)$ $\forall i, j, \ldots$ has also the right properties
- Correlations in k-space are often found with a bad random-number generators
- Another issue is the period of the generator
- The `ran()` function in `libc` has been (very) bad. Do not use this function in applications when good randomness is needed says `man 3 rand`

Random numer generators

Basic algorithm

- Many random number generators are based on the recurrence relation:

$$N_{j+1} = a \cdot N_j + c \pmod{m}$$

  These are called linear congruential generators. c is actually useless.
- "Divide" by $m + 1$ to get a number in the range $[0; 1[$
- Choices of a, m in standard libraries are found to range from very bad to relatively good
- A "minimal standard" set is $a = 7^5 = 16807$, $c = 0$, $m = 2^{31} - 1 = 2147483647$. This is RAN0
- Note that the period is at most m

Random numer generators

Improvements on RAN0

1. Multiplication by a doesn't span the whole range of values, i.e. if $N_i = 10^{-6}$, $N_{i+1} \leq 0.016$, failing a simple statistical test
   - Swap consecutive output values: Generate a few values (~ 32), and at each new call pick one at random. This is RAN1
2. The period $m = 2^{31} - 1$ might be too short
   - Add the outcome of two RAN1 generators with (slightly) different m's (and a's). The period is the least common multiple of m_1 and $m_2 \sim 2 \cdot 10^{18}$. This is RAN2
3. The generator is too slow
   - Use in C inline $N_{i+1} = 1664525 \cdot N_i + 1013904223$ using `unsigned long`. Patch the bits into a real number (machine dependent). This is RANQD2

Random numer generators

Implementations and recommendations

NR: Numerical Recipes
GSL: GNU Scientific Library

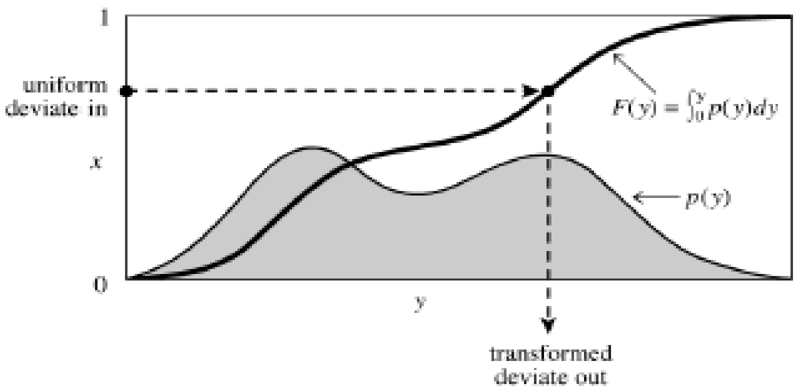
Library	Generator	Relative speed	Period
NR	RAN0	1.0	$\sim 2^{31}$
NR	RAN1	1.3	$\sim 2^{36}$
NR	RAN2	2.0	$\sim 2^{62}$
NR	RANQD2	0.25	$\sim 2^{30}$
GSL	MT19937	0.8	$\sim 2^{19937}$
GSL	TAUS	0.6	$\sim 2^{88}$
GSL	RANLXD2	8.0	$\sim 2^{400}$

Always use GSL! See the GSL doc for the many more algorithms available

Transformation method

The method

The transformation method allows in principle to draw values at random from any distribution



1. Given a distribution $p(y)$, the cumulative distribution function (CDF) of $p(y)$ is $F(y) = \int_0^y p(w)\,\mathrm{d}w$
2. We want to draw y uniformly in the shaded area, i.e. uniformly over $F(y)$; by construction $0 \leq F(y) \leq 1$,
3. We draw $x \sim \mathcal{U}(0,1)$ and find y so that $x = F(y)$
4. Therefore $y(x) = F^{-1}(x),\ x \sim \mathcal{U}(0,1)$

Transformation method

Example

Exponential deviates: $p(y) = \lambda e^{-\lambda y}$

$$F(y) = 1 - e^{-\lambda y} = x$$

$$y(x) = -\frac{1}{\lambda}\ln(1 - x)$$

Note: this is equivalent to

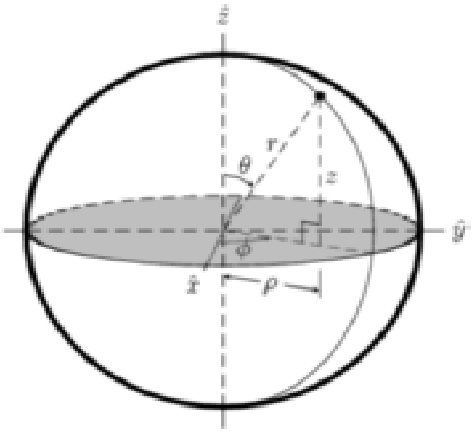
$$y(x) = -\frac{1}{\lambda}\ln(x),$$

since, if $x \sim \mathcal{U}(0, 1)$, then $1 - x \sim \mathcal{U}(0, 1)$ as well

Note also that it is rather uncommon to be able to calculate $F^{-1}(x)$ analytically. Depending on accuracy, it is possible to calculate an numerical approximation

Transformation method

A point in space



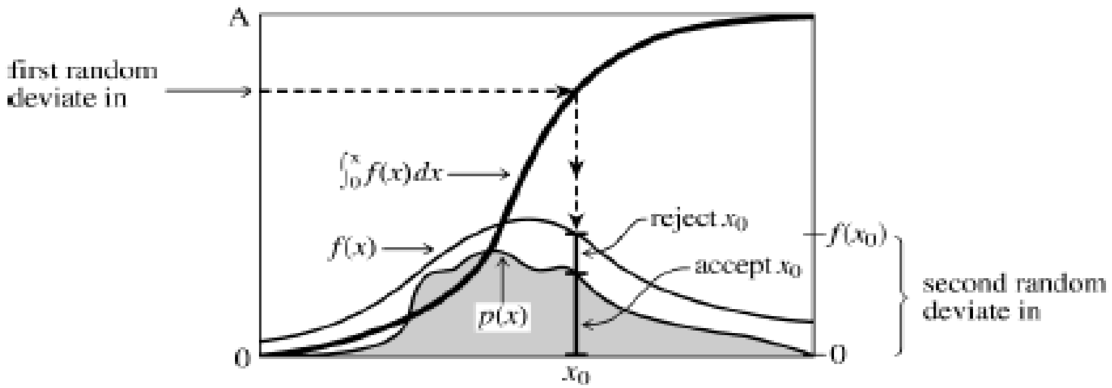
To draw a point in a homogeneous sphere of radius R:

1. ϕ can be drawn uniformly from $\mathcal{U}(0, 2\pi)$
2. θ has a sine distribution $p(\theta) = sin(\theta)/2,\ \theta \in [0; \pi[$
   Transformation: $\theta = 2\arccos(x)$
3. Each radius shell has a volume $f(R) \sim R^2\, \mathrm{d}R$, so
   $R \propto \sqrt[3]{x}$
4. Alternatively, draw a point at random on the surface of a sphere $(x, y, z)/\sqrt{x^2 + y^2 + z^2}$ with
   $x,\ y,\ z \sim \mathcal{N}(0, 1)$

Rejection method

The method

If the CDF of $p(x)$ is difficult to estimate (and you can forget about inversion), the rejection method can be used

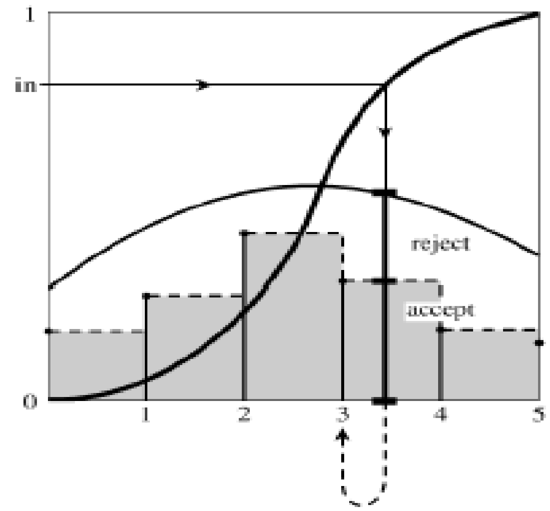


1. Find a comparison function $f(x)$ that can be sampled, so that $f(x) \geq p(x), \forall x$
2. Draw a random deviate x_0 from $f(x)$
3. Draw a uniform random deviate y_0 from $\mathcal{U}(0, f(x_0))$
4. If $y_0 < p(x_0)$, accept x_0, otherwise discard it
5. Repeat 2.–4. until you have enough values

The rejection method can be very inefficient if $f(x)$ is very different from $p(x)$

Rejection method

Example



The Poisson distribution is discrete: $\mathcal{P}(n;\alpha) = \frac{\alpha^n\ e^{-\alpha}}{n!}$

Make it continuous:

$$\mathcal{P}(x;\alpha) = \frac{\alpha^{[x]}\ e^{-\alpha}}{[x]!}$$

A Lorentzian $f(x) \propto \frac{1}{(x-\alpha)^2+c^2}$ is a good comparison function

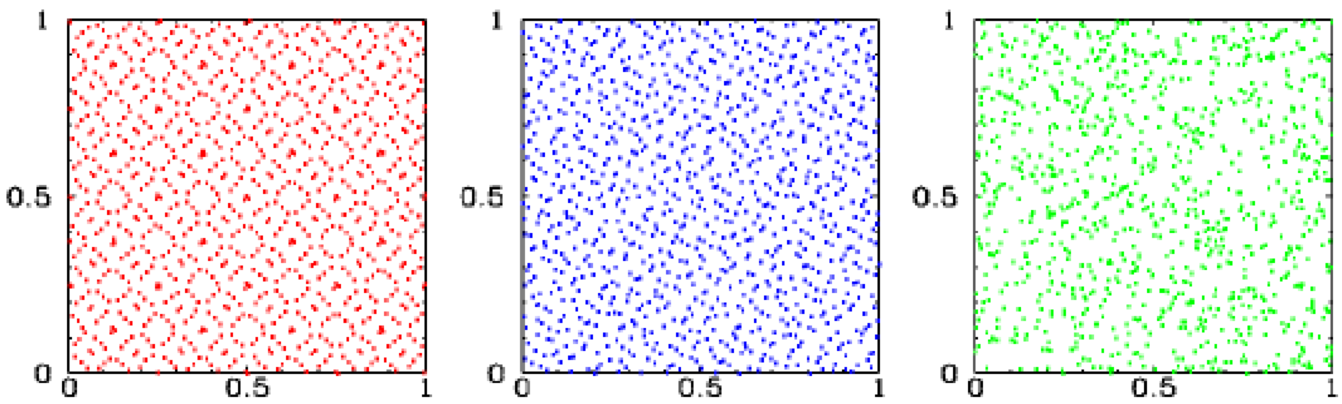
Distributions

GNU Scientific Library implements (not exhaustive!):

Gaussian	Binomial
Correlated bivariate Gaussian	Poisson
Exponential	
Laplace	
Cauchy	Spherical 2D, 3D
Rayleigh	
Landau	
Log-normal	
Gamma, beta	
χ^2, F, t	

Quasi-random numbers

What is random?



All sets of points fill "randomly" the area [[0; 1]; [0; 1]]

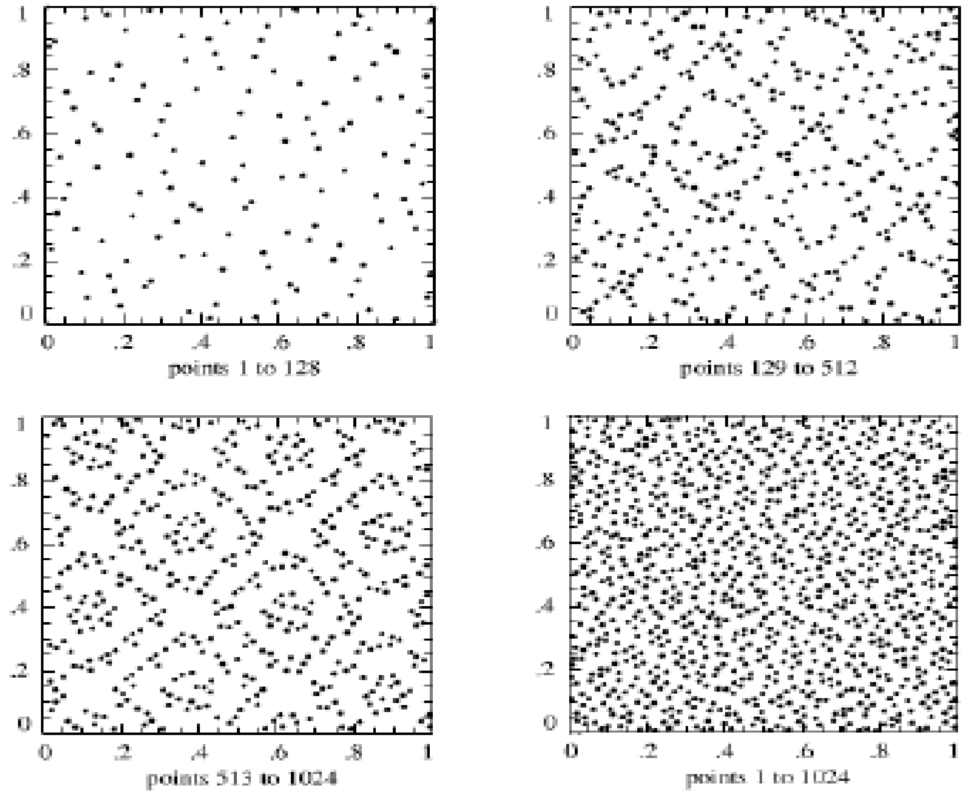
The left and center images are "sub-random" and fill more uniformly the area

These sequences are also called low-discrepancy sequences

These sequences can be used to replace the RNG when $x \sim \mathcal{U}(a, b)$ is needed

Quasi-random numbers

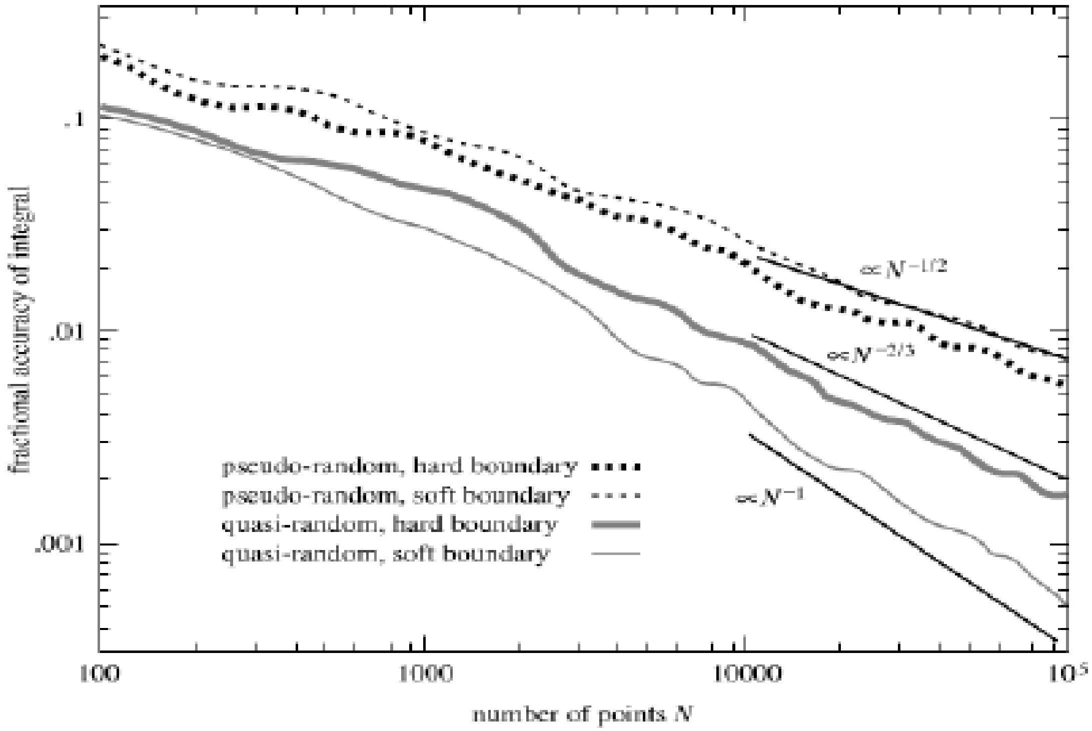
Filling of the plane



The sequence fills more or less uniformly the plane $\forall N$

Quasi-random numbers

Accuracy



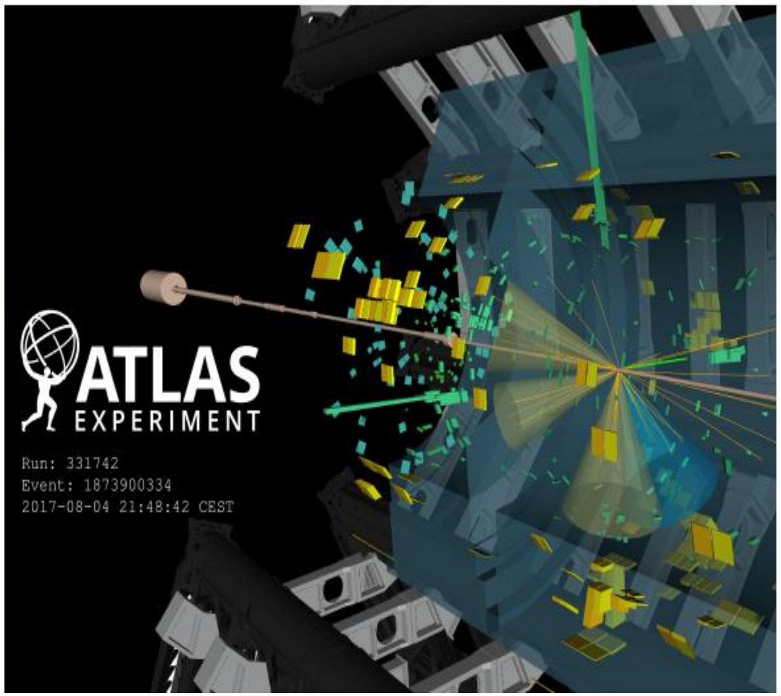
Convergence in some cases of numerical integration can reach $\sim 1/N$

Big picture: turning collision into publication

- **What we want:** statements about physical parameters ϕ, given data x_i collected by an experiment
  - connection: the likelihood $L_x(\phi) = p(x \mid \phi)$ — key ingredient for all subsequent statistical inference
  - $p(x \mid \phi)$ means: pick a ϕ and you get a probability density function over x

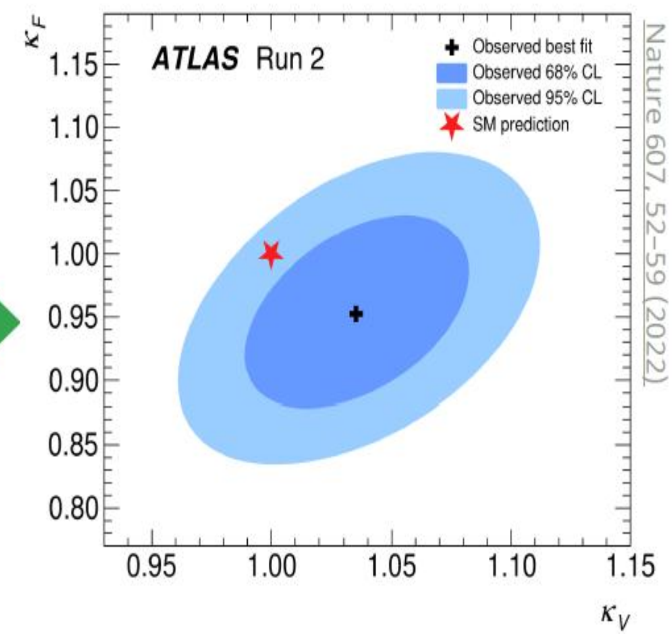
observations x_i

Phys. Lett. B 784 (2018) 173



$p(x \mid \phi)$

statements about parameters ϕ



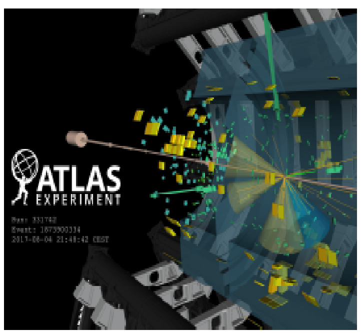
Nature 607, 52–59 (2022)

An antractable likelihood function

- We **need** $p(x \mid \phi)$ — unfortunately this very high-dimensional **integral** is ***intractable***, **cannot evaluate** this

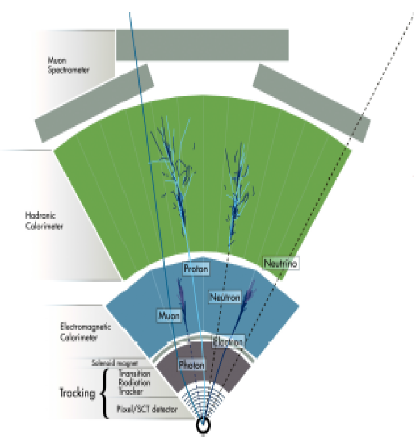
$$p\left(x \mid \phi\right) = \int \mathrm{d}z_D \mathrm{d}z_S \mathrm{d}z_P \; p\left(x \mid z_D\right) p\left(z_D \mid z_S\right) p\left(z_S \mid z_P\right) p\left(z_P \mid \phi\right)$$

observables x | **detector interaction** z_D | **parton shower** z_S | **parton level** z_P



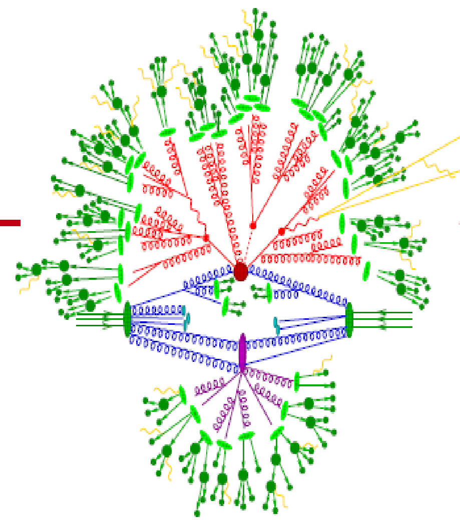
Phys. Lett. B 784 (2018) 173

←



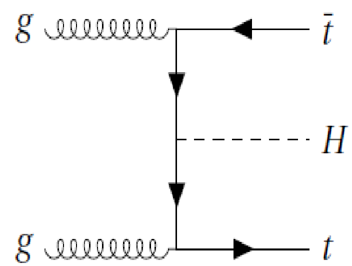
CERN-EX-1301009

←



JHEP 0902 (2009) 007

←

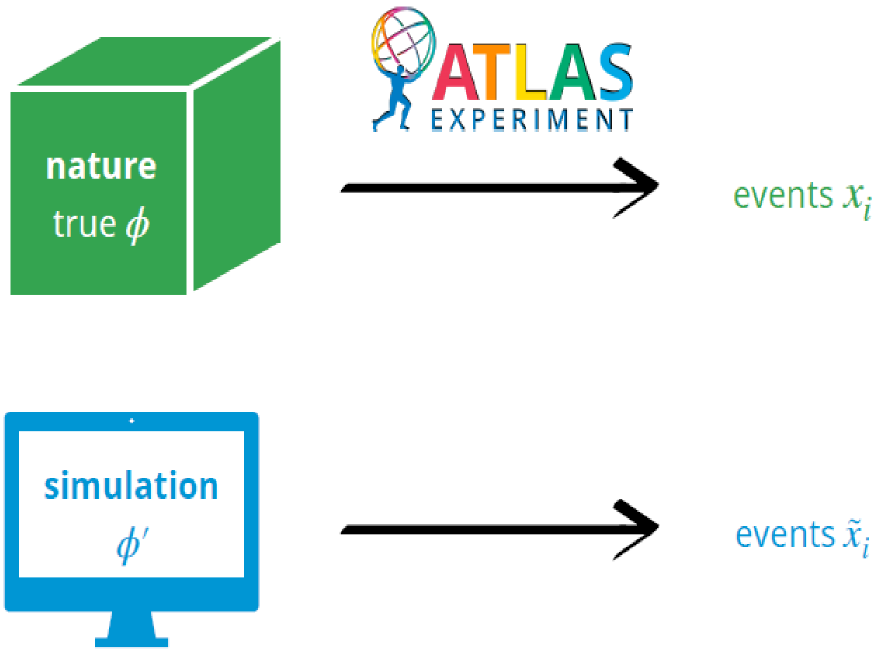


The dependence on parameters ϕ is here.

Simulation to approximate nature

- We wrote down $p(x \mid \phi)$, yet **cannot evaluate** it directly

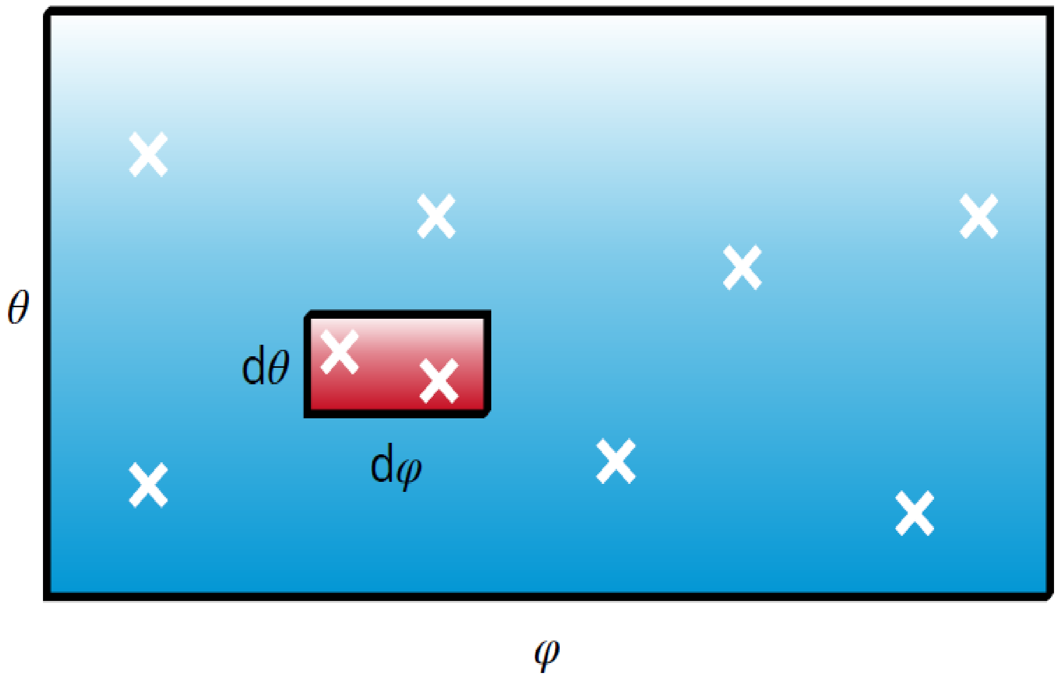
- Have a **set of simulators** for all steps involved and **can draw samples** $\tilde{x}_i \sim p(x \mid \phi')$, which **approximate nature**
  - another way to say this: we *can* "run Monte Carlo"



Simulation-based density estimation

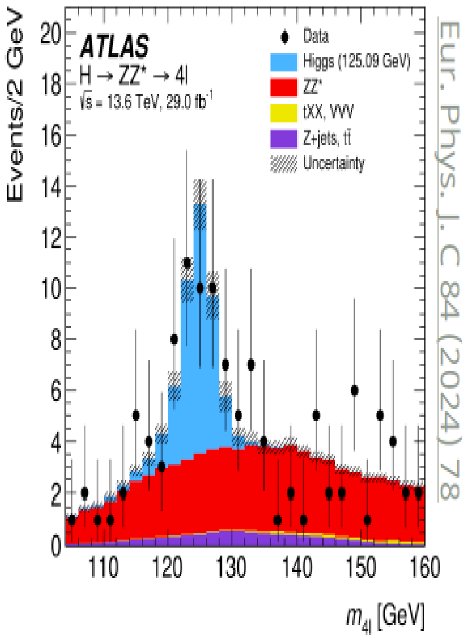
- Given **simulated events** $\tilde{x}_i \sim p(x \mid \phi')$ we can **construct the density** $p(\tilde{x} \mid \phi')$
  - this is an approximation of what we are after, the true $p(x \mid \phi)$

- Think of this as **MC integration**: with enough simulated events can construct approximate probability density

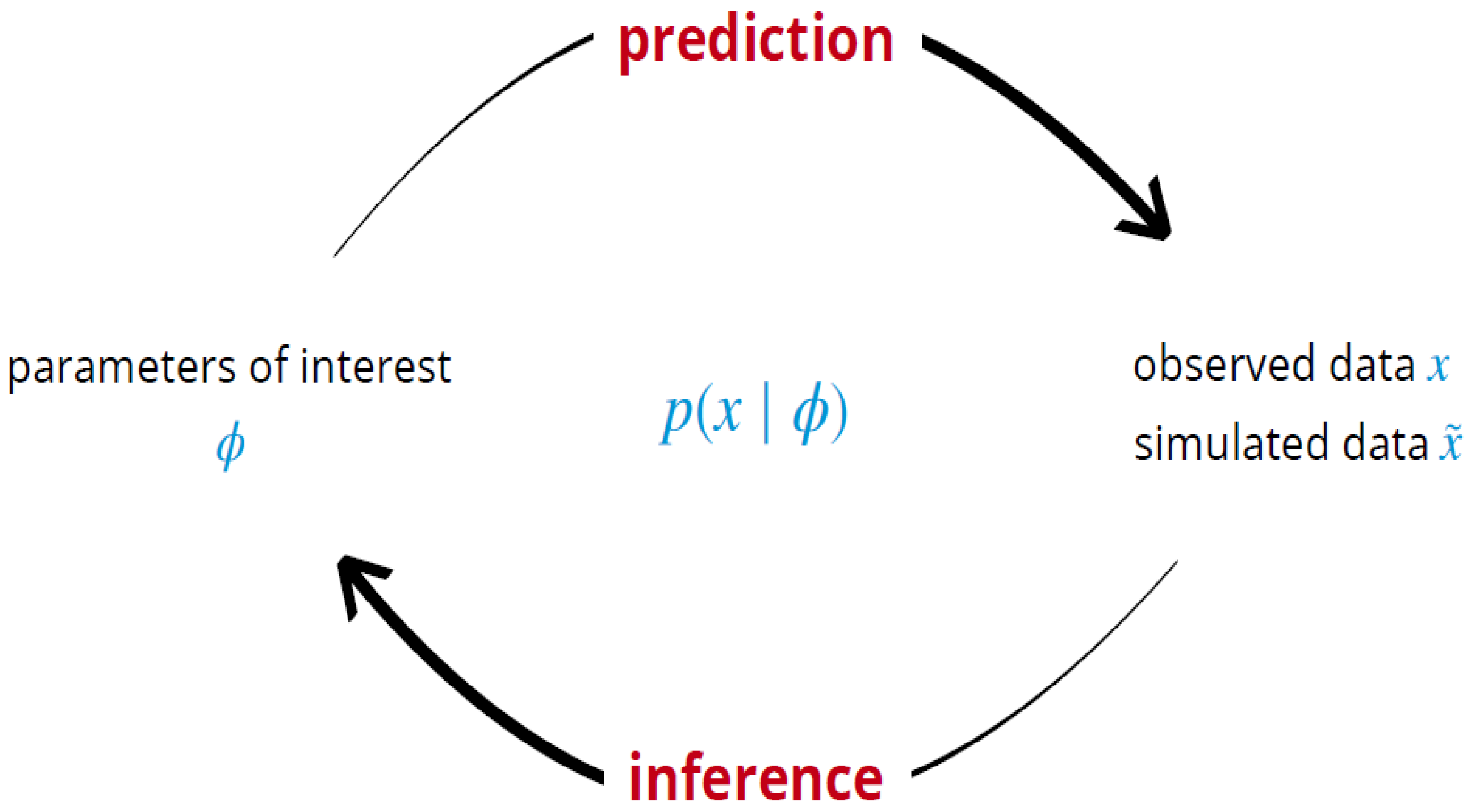


Histograms & summary statistics

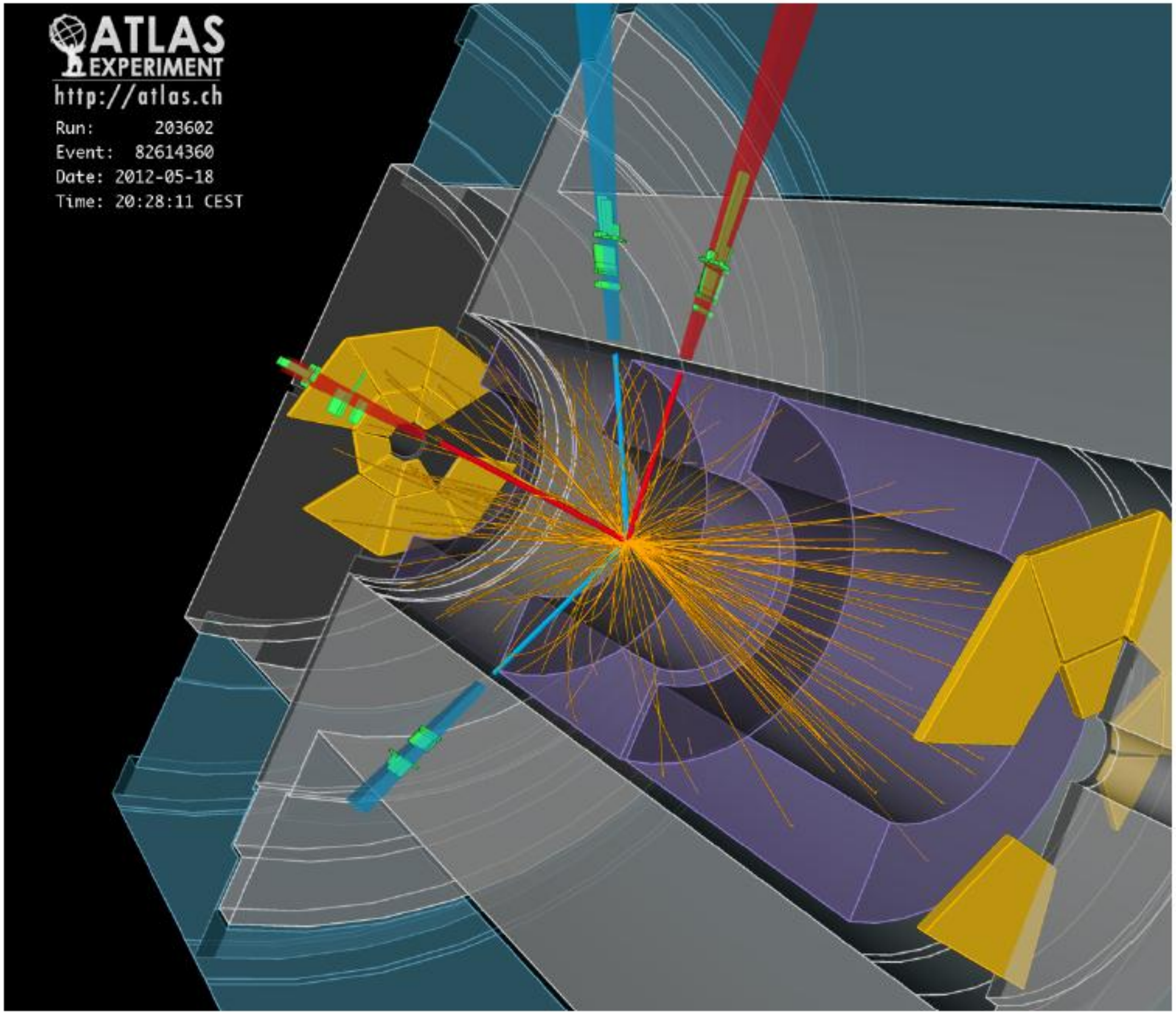
- Use MC samples to **estimate the density** $p(x \mid \phi)$, e.g. by **filling histograms** with the samples x_i
  - histograms are a convenient method for density estimation
- Histograms are hit by the **curse of dimensionality**
  - number of samples x_i needed scales exponentially with dimension of observation
- We use **summary statistics** to reduce dimensionality of our measurements
  - operate on objects like jets instead of detector channel responses
  - use physicists & machine learning to efficiently compress information
- **Challenge:** finding the right low-dimensional summary statistic — crucial for sensitivity



The statistical framing



LHC collision event



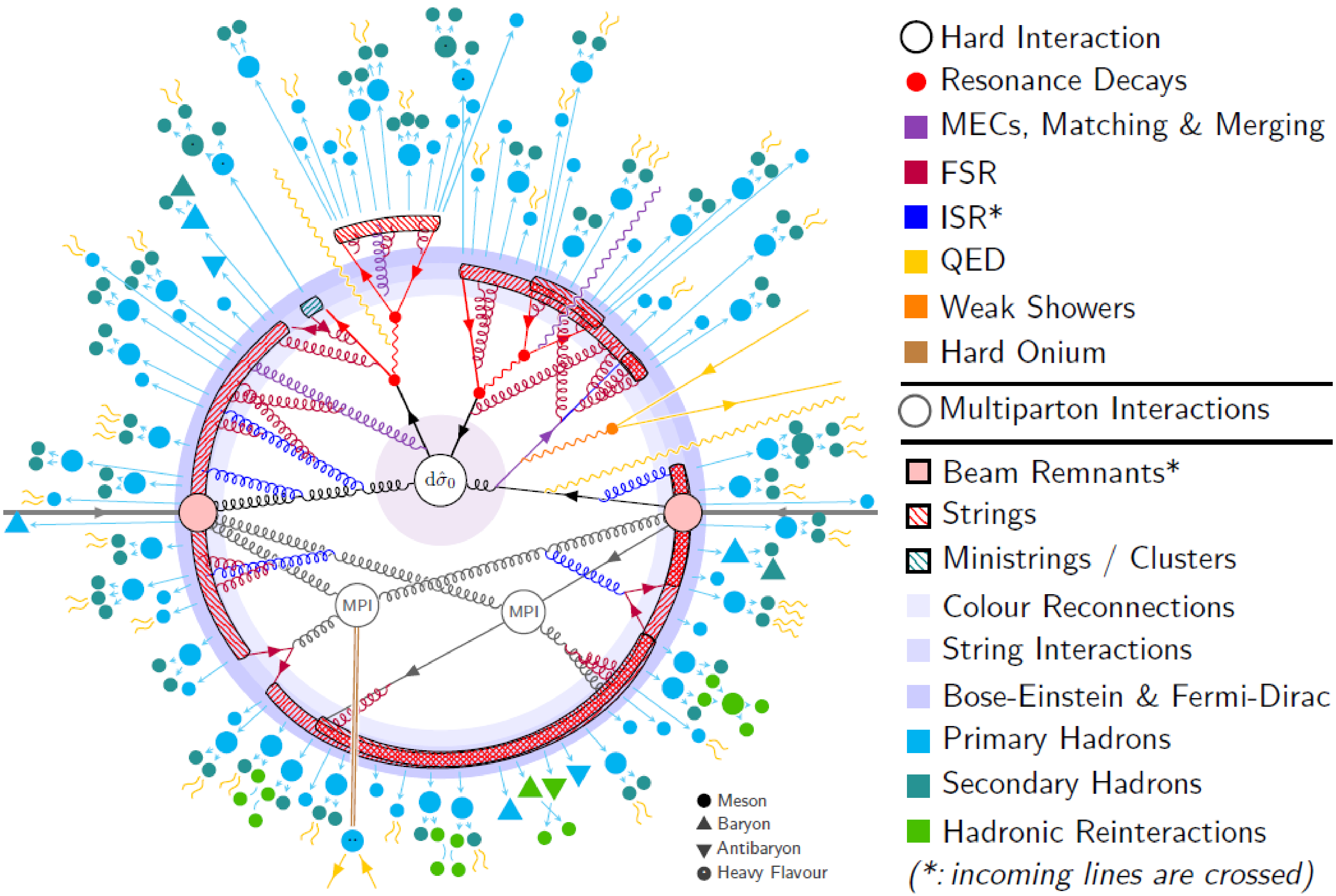
Four leptons clearly visible.

Maybe $H \to Z^0 Z^0 \to e^+ e^- \mu^+ \mu^-$.

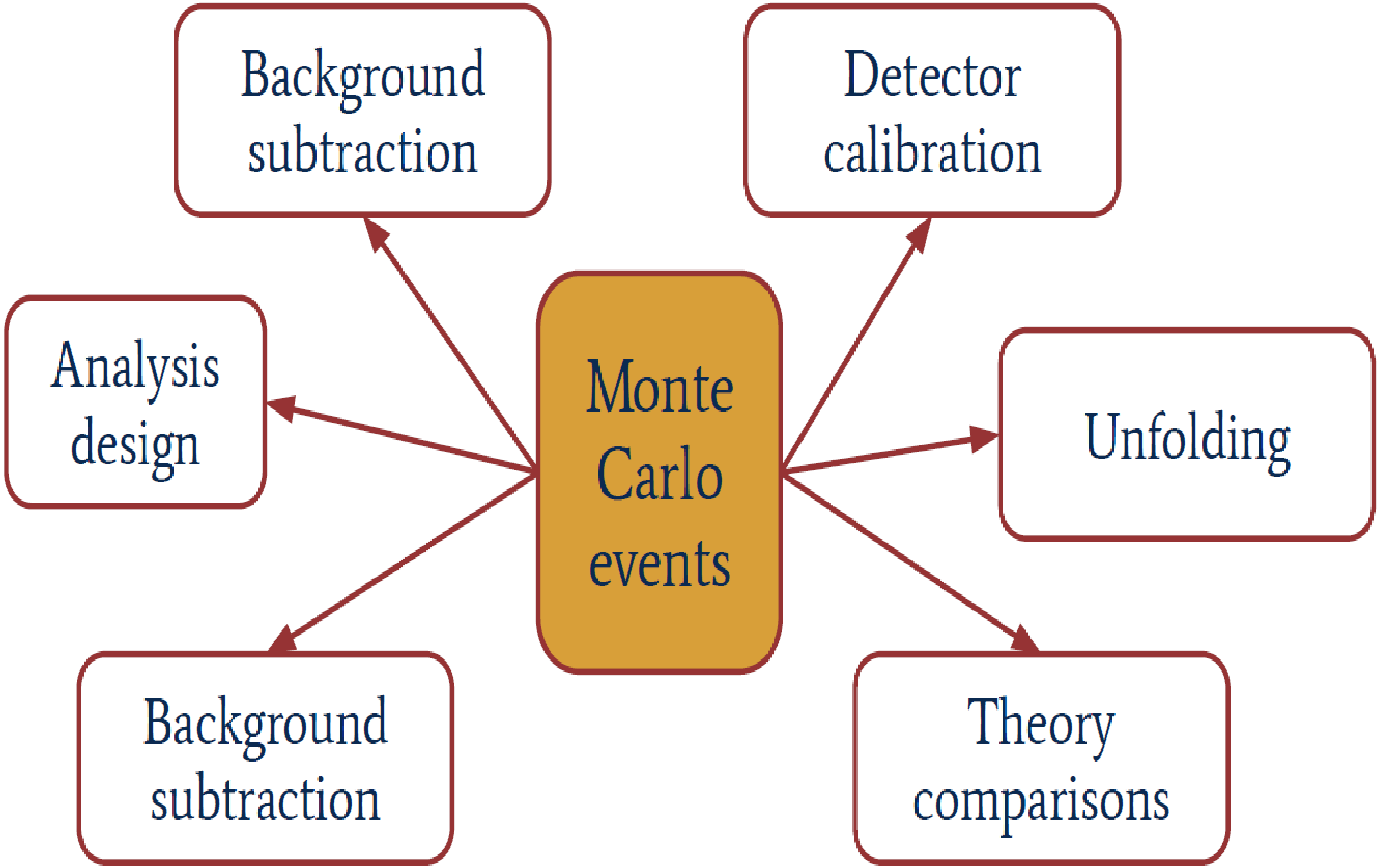
But what about rest of tracks?

Why and how are they produced?

A collected event view



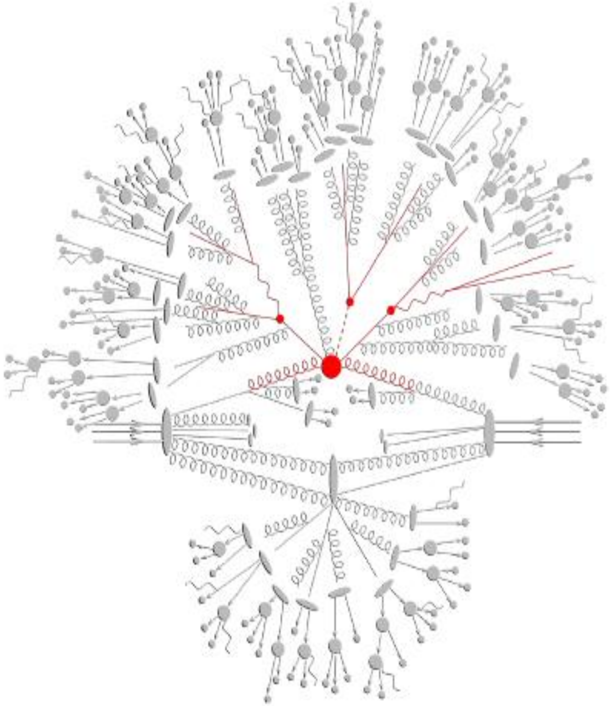
Monte Carlo events



General 2->n scattering cross-section

$$\hat{\sigma}_N = \int_{\text{cuts}} \mathrm{d}\hat{\sigma}_N = \int_{\text{cuts}} \left[\prod_{i=1}^{N} \frac{\mathrm{d}^3 q_i}{(2\pi)^3 2E_i} \right] \delta^4 \left(p_1 + p_2 - \sum_i^N q_i \right) |\mathcal{M}(p_1, p_2, q_1, \ldots, q_N)|^2$$

- Hard scattering matrix element
- Phase space integration including cuts



Monte Carlo task

1 Numerical integration for total cross section

- Needs MC methods due to high dimensionality $D \gtrsim 4$

2 Event generation

- $(3 \cdot N - 4)$ random numbers

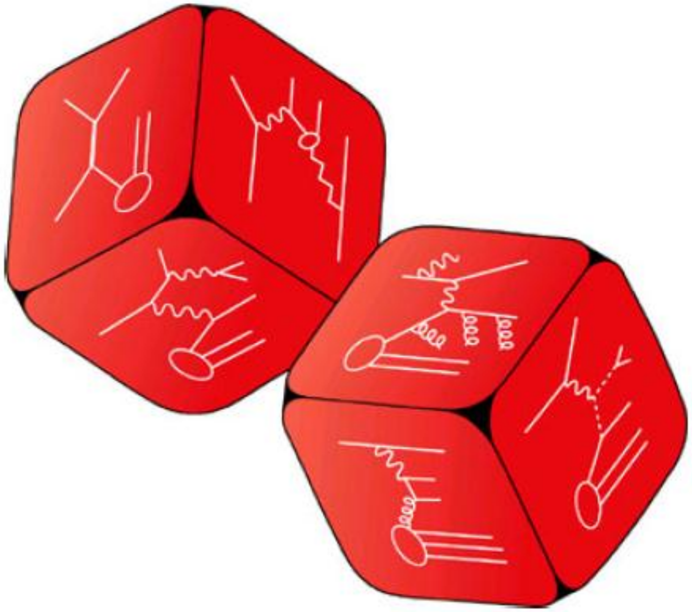
$\rightarrow$ N final state momenta

$\rightarrow$ natural "event" for $2 \rightarrow n$ scattering

$\Rightarrow$ Simply histogram any observable of interest

$\Rightarrow$ No need for dedicated calculations for observable

A tour to Monte Carlo



. . . because Einstein was wrong: God does throw dice!
Quantum mechanics: amplitudes $\Longrightarrow$ probabilities
Anything that possibly can happen, will! (but more or less often)

Event generators: trace evolution of event structure.
Random numbers $\approx$ quantum mechanical choices.

The Monte Carlo method

Want to generate events in as much detail as Mother Nature
$\Longrightarrow$ get average *and* fluctutations right
$\Longrightarrow$ make random choices, $\sim$ as in nature

$$\sigma_{\mathrm{final\ state}} = \sigma_{\mathrm{hard\ process}} \mathcal{P}_{\mathrm{tot,hard\ process}\to\mathrm{final\ state}}$$

(appropriately summed & integrated over non-distinguished final states)

where $\mathcal{P}_{\mathrm{tot}} = \mathcal{P}_{\mathrm{res}} \mathcal{P}_{\mathrm{ISR}} \mathcal{P}_{\mathrm{FSR}} \mathcal{P}_{\mathrm{MPI}} \mathcal{P}_{\mathrm{remnants}} \mathcal{P}_{\mathrm{hadronization}} \mathcal{P}_{\mathrm{decays}}$

with $\mathcal{P}_i = \prod_j \mathcal{P}_{ij} = \prod_j \prod_k \mathcal{P}_{ijk} = \ldots$ in its turn

$\Longrightarrow$ **divide and conquer**

an event with n particles involves $\mathcal{O}(10n)$ random choices,
(flavour, mass, momentum, spin, production vertex, lifetime, ...)
LHC: $\sim$ 100 charged and $\sim$ 200 neutral (+ intermediate stages)
$\Longrightarrow$ several thousand choices
(of $\mathcal{O}(100)$ different kinds)

Why generators?

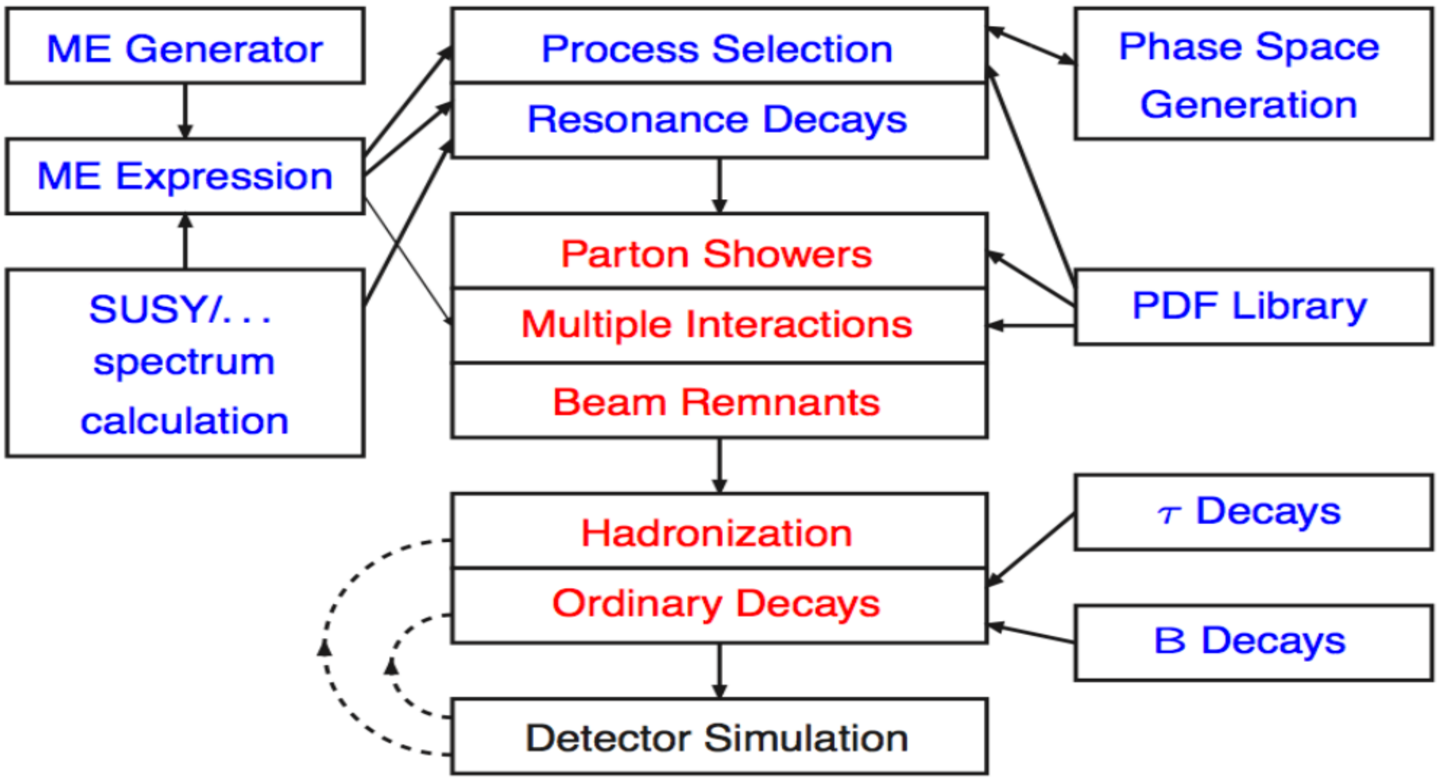
- Allow theoretical and experimental studies of *complex* multiparticle physics
- Large flexibility in physical quantities that can be addressed
- Vehicle of ideology to disseminate ideas from theorists to experimentalists

Can be used to

- predict event rates and topologies
  $\Rightarrow$ can estimate feasibility
- simulate possible backgrounds
  $\Rightarrow$ can devise analysis strategies
- study detector requirements
  $\Rightarrow$ can optimize detector/trigger design
- study detector imperfections
  $\Rightarrow$ can evaluate acceptance corrections

Few generic ones: Pythia, Sherpa, Herwig + other relevant packages

Put together for maximum effect



Standardized interfaces essential!

PDG particle codes

A. Fundamental objects

1	d	11	e^-	21	g	32	Z'^0	39	G
2	u	12	ν_e	22	γ	33	Z''^0	41	R^0
3	s	13	μ^-	23	Z^0	34	W'^+	42	LQ
4	c	14	ν_μ	24	W^+	35	H^0	51	DM_0
5	b	15	τ^-	25	h^0	36	A^0		
6	t	16	ν_τ			37	H^+	...	...

add − sign for antiparticle, where appropriate

+ diquarks, SUSY, technicolor, ...

B. Mesons

$100\,|q_1| + 10\,|q_2| + (2s+1)$ with $|q_1| \geq |q_2|$

particle if heaviest quark u, $\overline{s}$, c, $\overline{b}$; else antiparticle

111	π^0	311	K^0	130	K^0_L	221	η^0	411	D^+	431	D^+_s
211	π^+	321	K^+	310	K^0_S	331	η'^0	421	D^0	443	J/ψ

C. Baryons

$1000\,q_1 + 100\,q_2 + 10\,q_3 + (2s+1)$

with $q_1 \geq q_2 \geq q_3$, or Λ-like $q_1 \geq q_3 \geq q_2$

2112	n	3122	Λ^0	2224	Δ^{++}	3214	Σ^{*0}
2212	p	3212	Σ^0	1114	Δ^-	3334	Ω^-

Les Houches LHA/LHEF event record

At initialization:

- beam kinds and E's
- PDF sets selected
- weighting strategy
- number of processes

Per process in initialization:

- integrated σ
- error on σ
- maximum $\mathrm{d}\sigma/\mathrm{d(PS)}$
- process label

Per event:

- number of particles
- process type
- event weight
- process scale
- α_{em}
- α_{s}
- (PDF information)

Per particle in event:

- PDG particle code
- status (decayed?)
- 2 mother indices
- colour & anticolour indices
- (p_x, p_y, p_z, E), m
- lifetime τ
- spin/polarization

Monte Carlo techniques

"Spatial" problems: no memory/ordering

1. Integrate a function
2. Pick a point at random according to a probability distribution

"Temporal" problems: has memory

1. Radioactive decay: probability for a radioactive nucleus to decay at time t, given that it was created at time 0

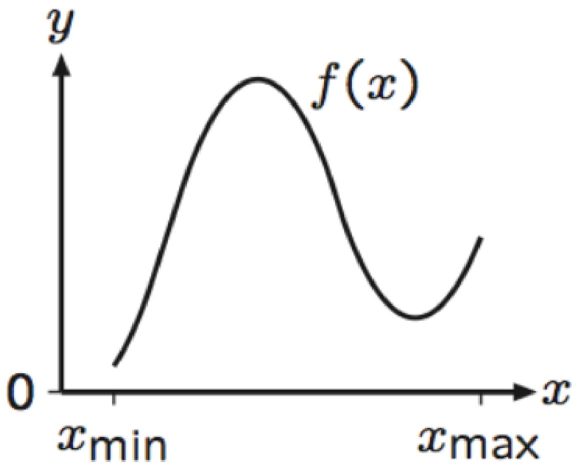
In reality combined into multidimensional problems:

1. Random walk (variable step length and direction)
2. Charged particle propagation through matter (stepwise loss of energy by a set of processes)
3. **Parton showers** (cascade of successive branchings)
4. Multiparticle interactions (ordered multiple subcollisions)

Assume algorithm that returns "random numbers" R, uniformly distributed in range $0 < R < 1$ and uncorrelated.

Integration and selection

Assume function $f(x)$,
studied range $x_{\min} < x < x_{\max}$,
where $f(x) \geq 0$ everywhere



Two connected standard tasks:

1 Calculate (approximatively)

$$\int_{x_{\min}}^{x_{\max}} f(x')\,\mathrm{d}x'$$

2 Select x at random according to $f(x)$

In step 2 $f(x)$ is viewed as "probability distribution"
with implicit normalization to unit area,
and then step 1 provides overall correct normalization.

Integral as an area/volume

Theorem

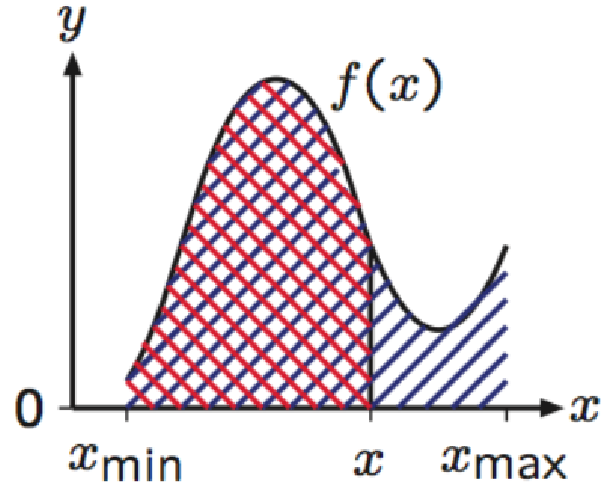
An n-dimensional integration $\equiv$ an $n+1$-dimensional volume

$$\int f(x_1,\ldots,x_n)\,\mathrm{d}x_1\ldots\mathrm{d}x_n \equiv \int\int_0^{f(x_1,\ldots,x_n)} 1\,\mathrm{d}x_1\ldots\mathrm{d}x_n\,\mathrm{d}x_{n+1}$$

since $\int_0^{f(x)} 1\,\mathrm{d}y = f(x)$.
So, for $1+1$ dimension, selection of x according to $f(x)$ is equivalent to uniform selection of (x,y) in the area $x_{\min} < x < x_{\max}$, $0 < y < f(x)$.
Therefore

$$\int_{x_{\min}}^{x} f(x')\,\mathrm{d}x' = R\int_{x_{\min}}^{x_{\max}} f(x')\,\mathrm{d}x'$$

(area to left of selected x is uniformly distributed fraction of whole area)



Analytical solution

If **know primitive function** $F(x)$ and **know inverse** $F^{-1}(y)$ then

$$\begin{aligned} F(x) - F(x_{\min}) &= R\,(F(x_{\max}) - F(x_{\min})) = R\,A_{\text{tot}} \\ \Longrightarrow x &= F^{-1}(F(x_{\min}) + R\,A_{\text{tot}}) \end{aligned}$$

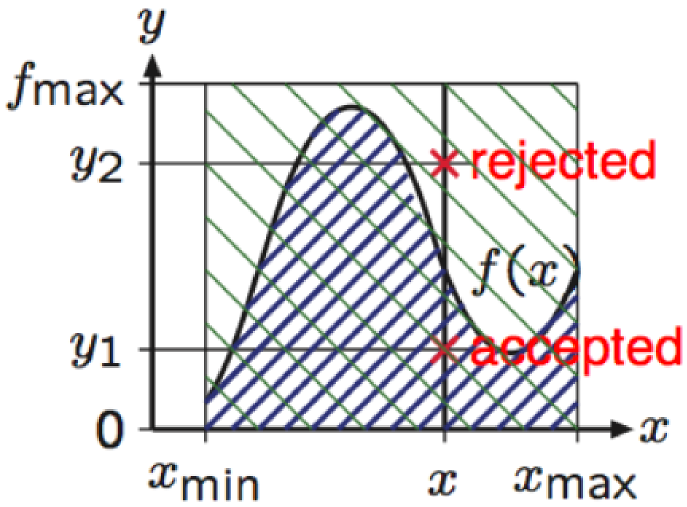
Proof: introduce $z = F(x_{\min}) + R\,A_{\text{tot}}$. Then

$$\frac{\mathrm{d}\mathcal{P}}{\mathrm{d}x} = \frac{\mathrm{d}\mathcal{P}}{\mathrm{d}R}\frac{\mathrm{d}R}{\mathrm{d}x} = 1\,\frac{1}{\frac{\mathrm{d}x}{\mathrm{d}R}} = \frac{1}{\frac{\mathrm{d}x}{\mathrm{d}z}\,\frac{\mathrm{d}z}{\mathrm{d}R}} = \frac{1}{\frac{\mathrm{d}F^{-1}(z)}{\mathrm{d}z}\,\frac{\mathrm{d}z}{\mathrm{d}R}} = \frac{\frac{\mathrm{d}F(x)}{\mathrm{d}x}}{\frac{\mathrm{d}z}{\mathrm{d}R}} = \frac{f(x)}{A_{\text{tot}}}$$

Hit-and-miss solution

If $f(x) \leq f_{\max}$ in $x_{\min} < x < x_{\max}$
use interpretation as an area

1. select $x = x_{\min} + R\,(x_{\max} - x_{\min})$
2. select $y = R\,f_{\max}$ (new R!)
3. while $y > f(x)$ cycle to 1



Integral as by-product:

$$I = \int_{x_{\min}}^{x_{\max}} f(x)\,\mathrm{d}x = f_{\max}\,(x_{\max} - x_{\min})\,\frac{N_{\mathrm{acc}}}{N_{\mathrm{try}}} = A_{\mathrm{tot}}\,\frac{N_{\mathrm{acc}}}{N_{\mathrm{try}}}$$

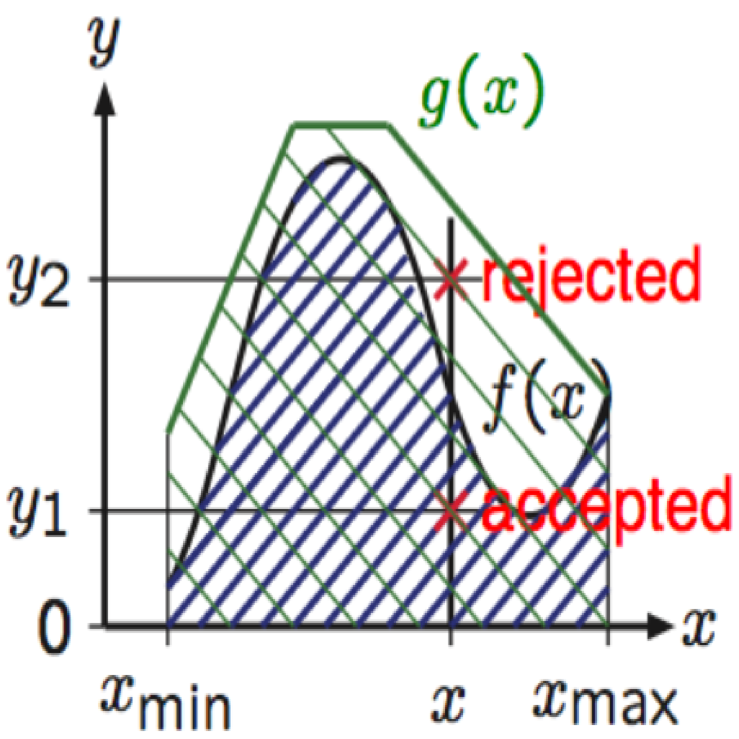
Binomial distribution with $p = N_{\mathrm{acc}}/N_{\mathrm{try}}$ and $q = N_{\mathrm{fail}}/N_{\mathrm{try}}$, so error

$$\frac{\delta I}{I} = \frac{A_{\mathrm{tot}}\sqrt{p\,q/N_{\mathrm{try}}}}{A_{\mathrm{tot}}\,p} = \sqrt{\frac{q}{p\,N_{\mathrm{try}}}} = \sqrt{\frac{q}{N_{\mathrm{acc}}}} < \frac{1}{\sqrt{N_{\mathrm{acc}}}}$$

Importance sampling

Improved version of hit-and-miss:
If $f(x) \leq g(x)$ in
$x_{\min} < x < x_{\max}$
and $G(x) = \int g(x')\,\mathrm{d}x'$ is simple
and $G^{-1}(y)$ is simple

1. select x according to $g(x)$ distribution
2. select $y = R\,g(x)$ (new R!)
3. while $y > f(x)$ cycle to 1



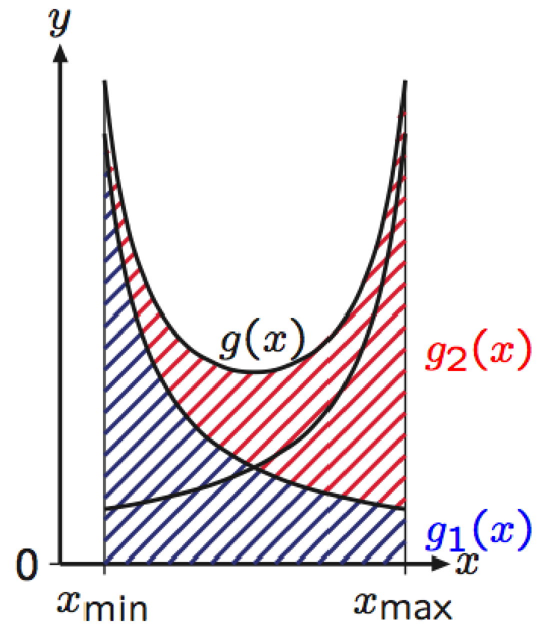
Multichannel

If $f(x) \leq g(x) = \sum_i g_i(x)$,
where all g_i "nice" ($G_i(x)$ invertible)
but $g(x)$ not

1. select i with relative probability

$$A_i = \int_{x_{\min}}^{x_{\max}} g_i(x')\,\mathrm{d}x'$$

2. select x according to $g_i(x)$
3. select $y = R\,g(x) = R\sum_i g_i(x)$
4. while $y > f(x)$ cycle to 1

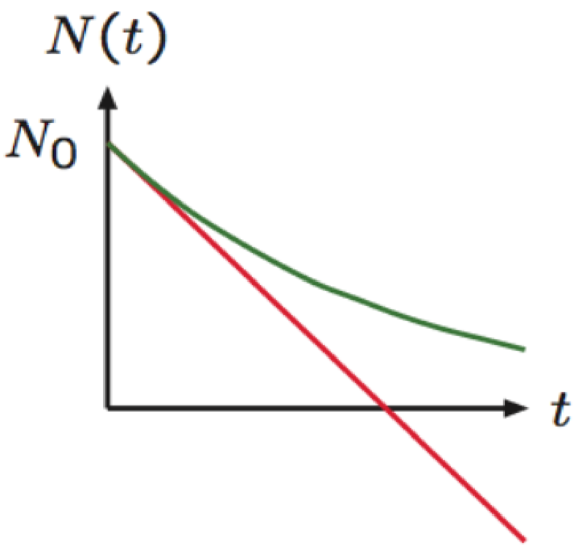


Works since

$$\int f(x)\,\mathrm{d}x = \int \frac{f(x)}{g(x)} \sum_i g_i(x)\,\mathrm{d}x = \sum_i A_i \int \frac{g_i(x)\,\mathrm{d}x}{A_i}\,\frac{f(x)}{g(x)}$$

Temporal methods: radioactive decays

Consider "radioactive decay":
$N(t)$ = number of remaining nuclei at time t
but normalized to $N(0) = N_0 = 1$ instead, so equivalently
$N(t)$ = probability that (single) nucleus has not decayed by time t
$P(t) = -\mathrm{d}N(t)/\mathrm{d}t$ = probability for it to decay at time t



Naively $P(t) = c \Longrightarrow N(t) = 1 - ct$.
Wrong! Conservation of probability driven by depletion:
a given nucleus can only decay once

Correctly
$P(t) = cN(t) \Longrightarrow N(t) = \exp(-ct)$
i.e. exponential dampening
$P(t) = c\exp(-ct)$

There is memory in time!

Temporal methods: radioactive decays

For radioactive decays $P(t) = cN(t)$, with c constant, but now generalize to time-dependence:

$$P(t) = -\frac{\mathrm{d}N(t)}{\mathrm{d}t} = f(t)\,N(t)\ ; \quad f(t) \geq 0$$

Standard solution:

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = -f(t)N(t) \iff \frac{\mathrm{d}N}{N} = \mathrm{d}(\ln N) = -f(t)\,\mathrm{d}t$$

$$\ln N(t) - \ln N(0) = -\int_0^t f(t')\,\mathrm{d}t' \implies N(t) = \exp\left(-\int_0^t f(t')\,\mathrm{d}t'\right)$$

$$F(t) = \int^t f(t')\,\mathrm{d}t' \implies N(t) = \exp\left(-(F(t) - F(0))\right)$$

Assuming $F(\infty) = \infty$, i.e. always decay, sooner or later:

$$N(t) = R \implies t = F^{-1}(F(0) - \ln R)$$

The veto algorithm: problem

What now if $f(t)$ has no simple $F(t)$ or F^{-1}?
Hit-and-miss not good enough, since for $f(t) \leq g(t)$, g "nice",

$$t = G^{-1}(G(0) - \ln R) \implies N(t) = \exp\left(-\int_0^t g(t')\,\mathrm{d}t'\right)$$

$$P(t) = -\frac{\mathrm{d}N(t)}{\mathrm{d}t} = g(t)\exp\left(-\int_0^t g(t')\,\mathrm{d}t'\right)$$

and hit-or-miss provides rejection factor $f(t)/g(t)$, so that

$$P(t) = f(t)\exp\left(-\int_0^t g(t')\,\mathrm{d}t'\right)$$

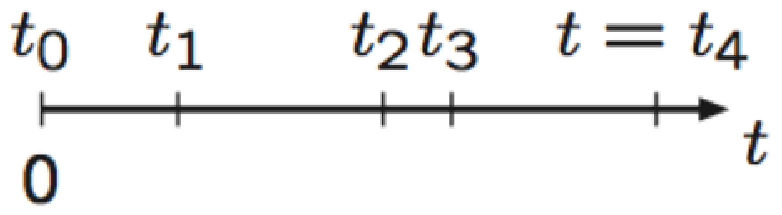
(modulo overall normalization), where it ought to have been

$$P(t) = f(t)\exp\left(-\int_0^t f(t')\,\mathrm{d}t'\right)$$

The veto algorithm: solution

The veto algorithm

1. start with $i = 0$ and $t_0 = 0$
2. $i = i + 1$
3. $t = t_i = G^{-1}(G(t_{i-1}) - \ln R)$, i.e $t_i > t_{i-1}$
4. $y = R\, g(t)$
5. while $y > f(t)$ cycle to 2



That is, when you fail, you keep on going from the time when you failed, and *do not* restart at time $t = 0$. (Memory!)

The winners take all

Assume "radioactive decay" with two possible decay channels 1&2

$$P(t) = -\frac{\mathrm{d}N(t)}{\mathrm{d}t} = f_1(t)N(t) + f_2(t)N(t)$$

Alternative 1:
use normal veto algorithm with $f(t) = f_1(t) + f_2(t)$.
Once t selected, pick decays 1 or 2 in proportions $f_1(t) : f_2(t)$.

Alternative 2:

The winner takes it all

select t_1 according to $P_1(t_1) = f_1(t_1)N_1(t_1)$
and t_2 according to $P_2(t_2) = f_2(t_2)N_2(t_2)$,
i.e. as if the other channel did not exist.
If $t_1 < t_2$ then pick decay 1, while if $t_2 < t_1$ pick decay 2.

Equivalent by simple proof.

Radioactive decay as perturbation theory

Assume we don't know about exponential function.
Recall wrong solution, starting from $N(t) = N_0(t) = 1$:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = -cN = -cN_0(t) = -c \Rightarrow N(t) = N_1(t) = 1 - ct$$

Now plug in $N_1(t)$, hoping to find better approximation:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = -cN_1(t) \Rightarrow N(t) = N_2(t) = 1 - c\int_0^t (1 - ct')\mathrm{d}t' = 1 - ct + \frac{(ct)^2}{2}$$

and generalize to

$$N_{i+1}(t) = 1 - c\int_0^t N_i(t')\,\mathrm{d}t' \Rightarrow N_{i+1}(t) = \sum_{k=0}^{i+1} \frac{(-ct)^k}{k!}$$

which recovers exponential e^{-ct} for $i \to \infty$.

Reminiscent of (QED, QCD) perturbation theory with $c \to \alpha f$.

Summary

Main event components:

- parton distributions
- hard subprocesses
- initial-state radiation
- final-state interactions
- multiparton interactions
- beam remnants
- hadronization
- decays
- total cross sections

Main Monte Carlo methods:

- integration as an area
- analytical solution
- hit-and-miss
- importance sampling
- multichannel
- **the veto algorithm**
- the winner takes it all