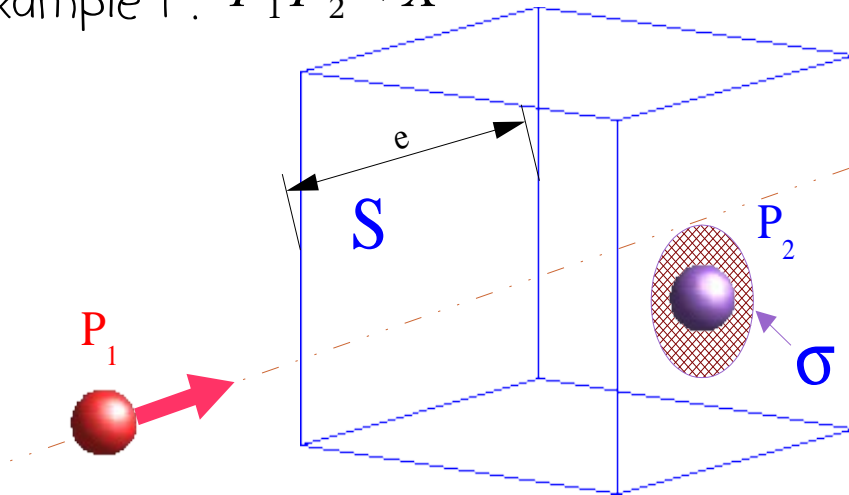


Error, Probability and Statistics

Définition

Detecting and counting subatomic particles is - by the very nature of the particle interaction laws - a quantum process. This means that statistical and probabilistic laws play a major role in the analysis and the interpretation of the subatomic measurements.

Example 1 : $P_1 P_2 \rightarrow X$



$$p = \frac{\sigma(P_1 P_2 \rightarrow X)}{S}$$

p reaction probability for 2 particles :
 p_1 impinging a box containing p_2

Probability density defined in the reaction phase space

$$\sigma(P_1 P_2 \rightarrow X) \propto \int_{\text{phase space}} |M|^2 d\Phi$$

Cross section of $P_1 P_2 \rightarrow X$ process

If the box is filled by an element :

$$P = p \frac{e S \rho}{M} N_A$$

specific mass (points to ρ)
 Avogadro number (points to N_A)
 Atomic mass (points to M)
 Probability of p_1 to interact with the box (points to P)

Example 2 :

Because of its quantum nature, an unstable nucleus or subatomic particle decays with a constant probability λ per unit time which is given by the inverse of its mean decay time :

$$\lambda dt = \frac{1}{\tau} dt$$

↙ mean decay time

If there's a collection of N_0 such objects at $t_0=0$: $dN(t) = -N(t) \frac{1}{\tau} dt$

If we count the number of decays starting at $t_0 = 0$ over a short period Δt compared to τ , we predict :

$$N(t) = N_0 e^{-t/\tau}$$

↙ Population at time t

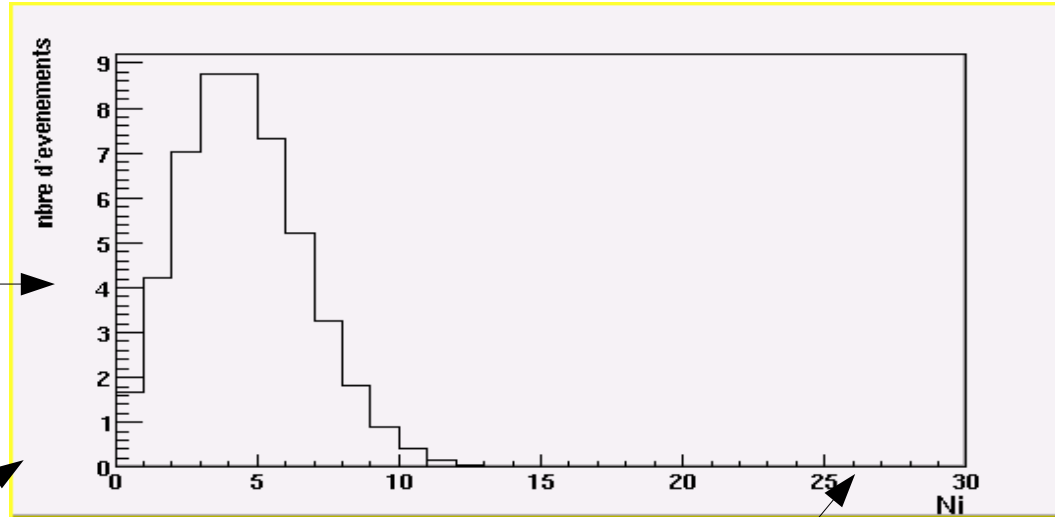
$$N_p = N_0 \frac{\Delta t}{\tau}$$

Suppose N_p is small, and we carry out N identical experiments counting the number of decays over Δt :

N_i is the integer number of decays observed in one of these experiments

The set of experiments is a **sample of size N** .

Y-axis : n_i
Number of occurrences of N_i decays



Such a picture is called a **histogram**

x-axis : number of decays observed

The **sampling frequency** is defined by :

$$f(N_i) = \frac{n_i}{N}$$

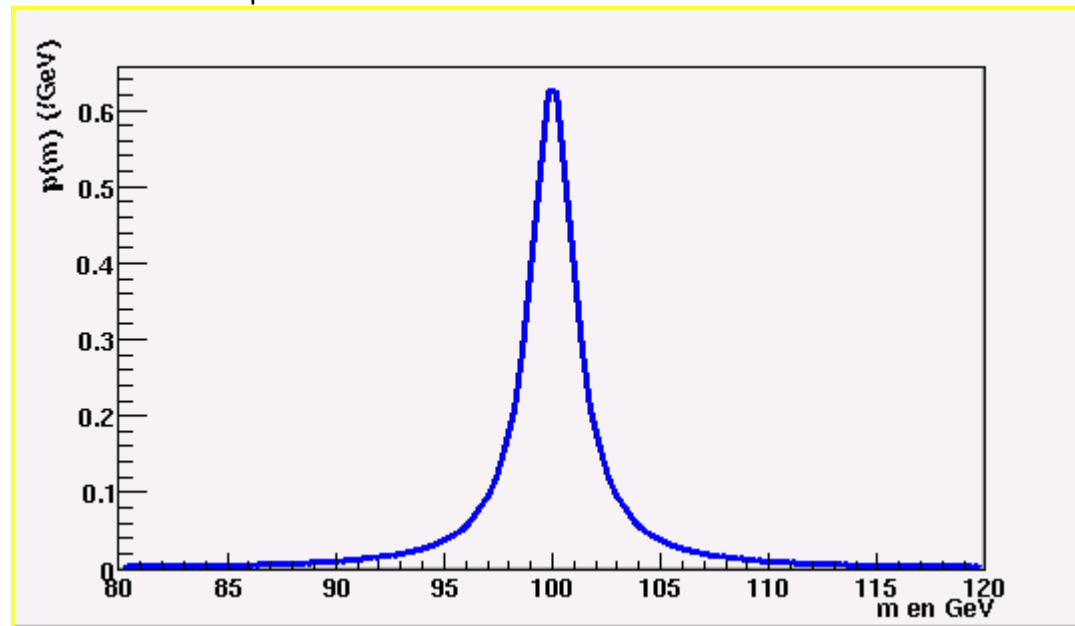
$$\sum_i f(N_i) = 1$$

Definition of probability :

$$p(N_i) = \lim_{N \rightarrow \infty} f(N_i) = \lim_{N \rightarrow \infty} \frac{n_i}{N}$$

N_i is a **discrete random variable**

There also exist *continuous random variables*, e.g. the invariant mass of an unstable subatomic particle.



Since it is never possible to measure a sample of infinite size, the properties of a physical random process can never be exactly measured. A measurement which is derived from a finite sample will always carry an uncertainty called *statistical error*.

In general, the *statistical error* decreases when the sample size increases.

On top of this statistical error, we often need to consider a **systematic (instrumental) error** which reflects the exactness of the measurement method. .

Example : the counting of Ni is done using a detector which features an efficiency $\epsilon \leq 1$.

$$N_p = \frac{\langle N_i \rangle}{\epsilon}$$

← mean value of N_i

Statistical error

Systematic error

Total error →

$$\frac{\Delta N_p}{N_p} = \frac{\Delta \langle N_i \rangle}{\langle N_i \rangle} + \frac{\Delta \epsilon}{\epsilon}$$

$$\frac{\Delta N_p}{N_p} \quad \text{total relative error on } N_p$$

$$\Delta N_p \quad \text{total absolute error on } N_p$$

Random variables

Probability density :

$$\int_{-\infty}^{+\infty} f(x) dx = 1 \quad \text{for a continuous variable } x$$

$$\sum_{i=1}^j f(x_i) = 1 \quad \text{for a discrete variable } x_i$$

Distribution function : probability of $x < a$

$$F(a) = \int_{-\infty}^a f(x) dx \quad \text{for a continuous variable } x$$

$$F(a) = \sum_{i=1}^n f(x_i) \quad \text{for a discrete variable } x_i \text{ with } x_n \leq a \text{ and } x_{n+1} > a$$

Mean :

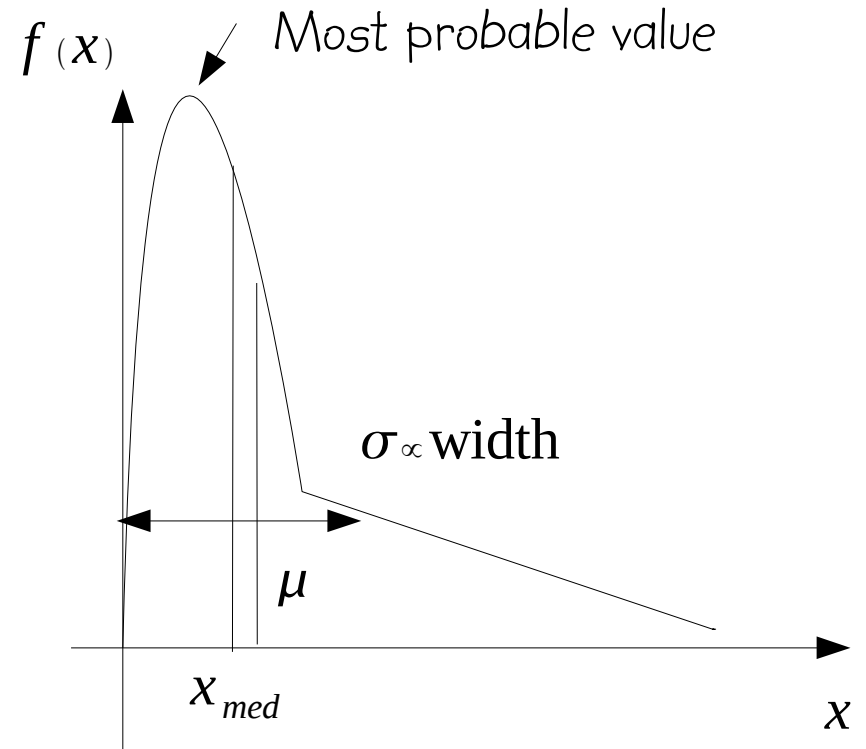
$$\mu = \int_{-\infty}^{+\infty} x f(x) dx = E(x)$$

Median :

$$F(x_{med}) = \int_{-\infty}^{x_{med}} f(x) dx = 0.5$$

Variance and standard deviation :

$$\begin{aligned} \text{var}(x) = \sigma^2(x) &= \int_{-\infty}^{+\infty} (x - \mu)^2 f(x) dx \\ &= E(x^2) - \mu^2 \end{aligned}$$



Probability density of two random variables :

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) dx dy = 1$$

Covariance of two random variables : measures the mutual dependence of x and y

$$E((x - \mu_x)(y - \mu_y)) = cov(x, y)$$

Correlation coefficient :

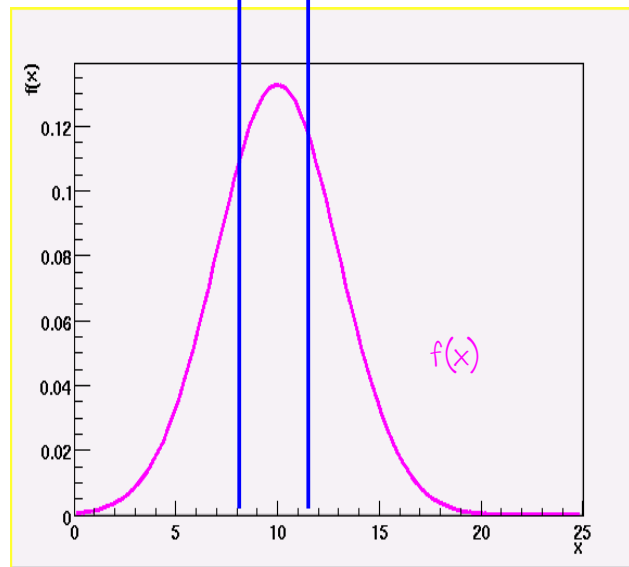
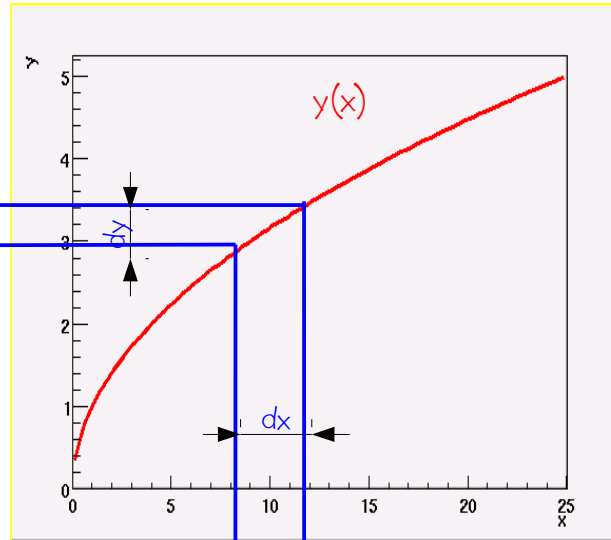
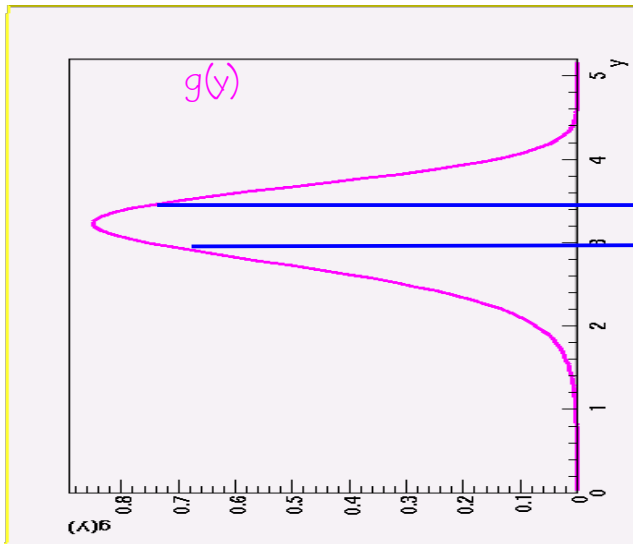
$$\rho_{xy} = \frac{cov(x, y)}{\sigma_x \sigma_y} \quad \text{with} \quad -1 \leq \rho_{xy} \leq 1 \quad \text{and} \quad \sigma_x, \sigma_y \quad \text{standard deviations of} \\ \text{x and y}$$

Independent random variables :

$$f(x, y) = f_1(x) f_2(y)$$

$$\text{then : } \rho_{xy} = cov(x, y) = 0$$

Change of variables :



$f(x)$ known probability density.

$y(x)$ is a monotonic function of x

y is a new random variable

$g(y)$?

$$g(y) dy = f(x) dx$$

$g(y), f(x) \geq 0$ then

$$g(y) = \left| \frac{dx}{dy} \right| f(x) = \left| \frac{dx}{dy} \right| f(w(y))$$

$x = w(y)$ inverse function of y

Binomial distribution :

n identical and independent objects which evolve according to a common binary random process (1 ou 0 : success or failure)

Probability of x successes among n trials

$$f(x, n, p) = C_n^x p^x q^{n-x} \quad p \text{ success probability, } q = 1 - p \text{ failure probability}$$

$$C_n^x = \frac{n!}{x!(n-x)!}$$

$$E(x) = \sum_{i=1}^n E(x_i) = n p \quad X_i \text{ being the } i\text{th trial}$$

$$\text{Var}(x) = \sum_{i=1}^n \text{var}(x_i) = \sum_{i=1}^n E((x_i - p)^2) = n((1-p)^2 p + (0-p)^2 q) = n p q$$

example : 100 neutron decays observed with a detector featuring an efficiency of 0.3

$$p = 0.3 \quad , \quad q = 0.7$$

$$E(x) = 100 \times 0.3 = 30$$

$$\sigma(x) = \sqrt{100 \times 0.3 \times 0.7} = 4.6$$

Poisson distribution :

If a binomial random process takes place among a very large number of objects ($n \rightarrow \text{infinity}$) and if its average (np) is finite, the binomial distribution reads :

$$f(x, \lambda) = \frac{\lambda^x}{x!} e^{-\lambda} \quad x \text{ is still a discrete random variable}$$

where $\lambda = np$ is the mean value $n \rightarrow \infty, p \rightarrow 0$
 $\lambda > 0$

means that a very large number of independent objects with a very small success probability

$$E(x) = \sigma^2(x) = \lambda$$

examples : the number of physicists sharing the Nobel prize each year is (almost) a Poisson distribution !

More seriously, the number of nuclei decaying in a radioactive source within a short time with respect to its period is a Poisson distribution.

Normal or Gaussian distribution :

It is another limit behavior of the binomial distribution when np (the mean value) becomes big.

$$f(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

x is a continuous random variable

$$E(x) = \mu$$

$$\text{var}(x) = \sigma^2$$

$$P(|x - \mu| \leq 1\sigma) = 0.682$$

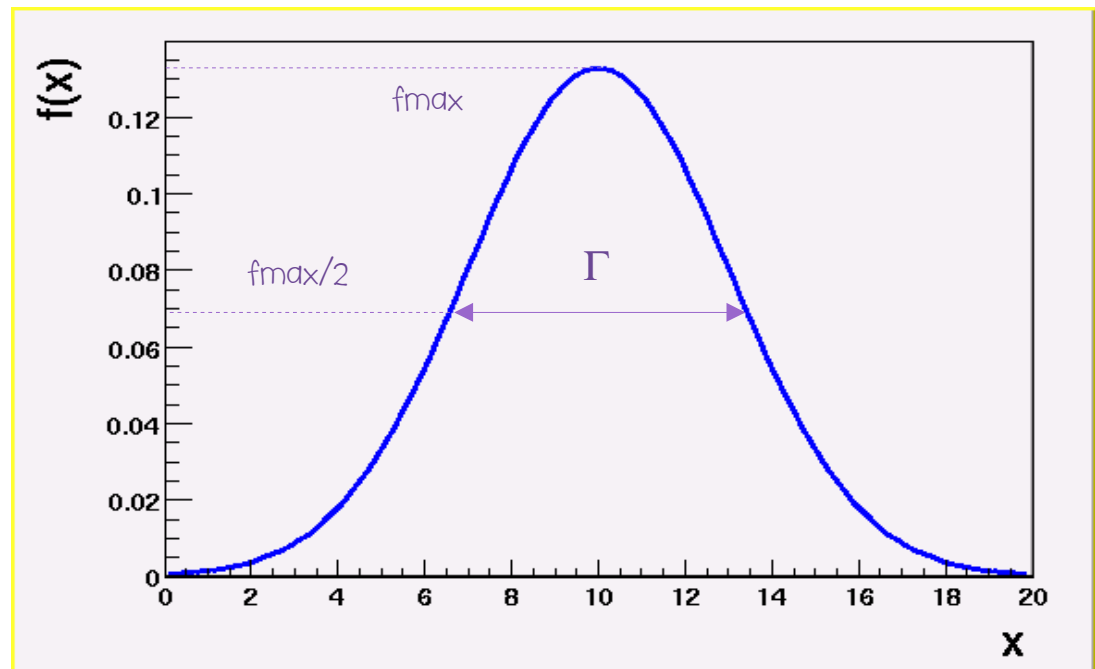
$$P(|x - \mu| \leq 2\sigma) = 0.954$$

$$P(|x - \mu| \leq 3\sigma) = 0.998$$

if x_1 et x_2 are two Gaussian and independent random numbers, then $x_1 + x_2$ is a Gaussian random number with :

$$\text{mean} : \mu = \mu_1 + \mu_2$$

$$\text{Standard deviation} : \sigma^2 = \sigma_1^2 + \sigma_2^2$$



Full Width at Half Maximum FWHM : $\Gamma = 2.354 \sigma$

χ^2 distribution :

if x_1, x_2, \dots, x_n are normal and independent random variables with means μ_i and standard deviations σ_i , then :

$\chi_n^2 = \sum_{i=1}^n \left(\frac{x_i - \mu_i}{\sigma_i} \right)^2$ which is a random variable, follows a χ^2 distribution with n degrees of freedom.

$$f(\chi_n^2, n) = \frac{(\chi_n^2)^{\frac{n}{2}-1} \exp\left(-\frac{\chi_n^2}{2}\right)}{2^{n/2} \Gamma\left(\frac{n}{2}\right)} \quad \text{Euler function : } \Gamma\left(\frac{n}{2}\right) = \int_0^{+\infty} t^{\frac{n}{2}-1} e^{-t} dt$$

$$E(\chi_n^2) = n, \quad \text{var}(\chi_n^2) = 2n$$

$\sum_i \chi_{n_i}^2$ follows a χ^2 distribution with $n = \sum_i n_i$ degrees of freedom

For a sample of n events x_i drawn from the same Gaussian random variable with mean μ and standard deviation σ , $\sum_{i=1}^n \left(\frac{x_i - \mu}{\sigma} \right)^2$ follows a chi-square distribution with n degrees of freedom

Central limit theorem :

This theorem states that the sum of n independent random variables x_i tends to a Gaussian when n becomes big. $x = \sum_i^n x_i$

$$E(x) = \sum_i^n \mu_i \qquad \text{var}(x) = \sum_{i=1}^n \sigma_i^2$$

Laplace model of experimental error (put forward in 1783) :

$$x = x_{true} + e_1 + e_2 + \dots + e_n = x_{true} + \sum_{i=1}^n e_i$$

with $E(e_i) = 0$ $\sigma(x_{true}) = 0$

if n is big , lots of error sources,
the sum is normally distributed .

This explains why experimental errors are so often normally distributed.

Considering that this model applies, very often measurement results are given according to the following convention : $m \pm \sigma$ at 68% confidence level

But be careful that the applicability of the central limit is never really demonstrated !
which means that the 68% confidence level may be optimistic.

Sampling and Estimators

A **sample** x_1, \dots, x_n of size n is a sub-set of a total population of a random variable.

A **statistics** is a function of a sample : $f(x_1, \dots, x_n)$.

An **estimator** is a statistics which is aimed at producing an estimated value of one of the parameters of the random variable probability density .

An **estimator is unbiased** if for all sample sizes, its average is equal to the sought probability density parameter.

Unbiased estimator of a mean :

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \qquad \sigma(\bar{x}) = \frac{\sigma}{\sqrt{n}}$$

Unbiased estimator of a variance :

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 = \frac{1}{n-1} \left(\sum_{i=1}^n x_i^2 - n\bar{x}^2 \right)$$

Estimator of counting :

Very often one counts the number of particles that satisfy some conditions (cuts on energy, angular position ...). This is a binomial random process.

If k is the number of selected events (satisfying conditions) among a population of size n , one may show that the statistical counting error of k is given by :

$$\sigma(k) = \sqrt{k \left(1 - \frac{k}{n}\right)}$$

Most of the times, when n is big compared to k , this reduces to :

$$\sigma(k) = \sqrt{k}$$

But be careful, if the counting efficiency is sizeable, this may not be valid anymore.

If k is big , then the counting law tends to a Gaussian of mean k and standard deviation given by the expression above.

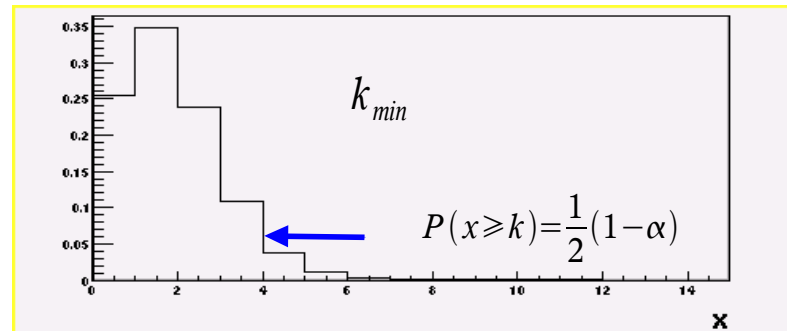
If k is small and $k/n \ll 1$

the counting law reduces to a Poisson distribution of mean k and standard deviation \sqrt{k}

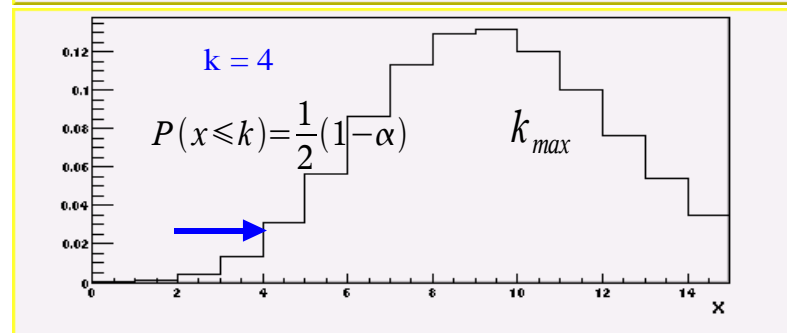
But because a Poisson distribution is asymmetric, the method to find a confidence interval for the counts is different.

To obtain a confidence interval on k at a α confidence level :

$$P(x \geq k, k_{min}) \leq \frac{1}{2}(1 - \alpha)$$



$$P(x \leq k, k_{max}) \leq \frac{1}{2}(1 - \alpha)$$



α for $k_{min} < k_{true}$	α for $k_{min} < k_{true} < k_{max}$	α for $k_{true} < k_{max}$	k_{min}	k	k_{max}
		84%	0	0	1,84
84%	68%	84%	0,17	1	3,3
84%	68%	84%	0,71	2	4,64
84%	68%	84%	1,37	3	5,92
84%	68%	84%	2,09	4	7,16
84%	68%	84%	3,62	6	9,58
84%	68%	84%	6,89	10	14,26
84%	68%	84%	15,57	20	25,54
84%	68%	84%	42,95	50	58,11
		95%	0	0	3
95%	90%	95%	0,05	1	4,74
95%	90%	95%	0,35	2	6,3
95%	90%	95%	0,82	3	7,75
95%	90%	95%	1,37	4	9,15
95%	90%	95%	2,61	6	11,84
95%	90%	95%	5,43	10	16,96
95%	90%	95%	13,26	20	29,06
95%	90%	95%	38,96	50	63,28

Least Square method :

case of direct measurement : sample of n independent measurements y_i of the same sought quantity x with measurement errors σ_i

$$M = \sum_{i=1}^n \left(\frac{y_i - x}{\sigma_i} \right)^2$$

which we minimize with respect to x

$$\frac{\partial M}{\partial x} = 0 \Rightarrow \bar{x} = \frac{\sum_{i=1}^n \frac{y_i}{\sigma_i^2}}{\sum_{i=1}^n \frac{1}{\sigma_i^2}}$$

Weighted mean of measurements with different errors

$\epsilon_i = y_i - \bar{x}$ are called residues, they are normally distributed

M follows a chi square law with $n-1$ degrees of freedom

Chi square Test of the result :

Given a confidence level α , one finds : χ_{max}^2 such that :

$$\int_0^{\chi_{max}^2} f(\chi^2, n-1) d\chi^2 = (1-\alpha) \quad \text{The results are given in quantile tables}$$

If : $M > \chi_{max}^2$ then the result x is rejected at a confidence level

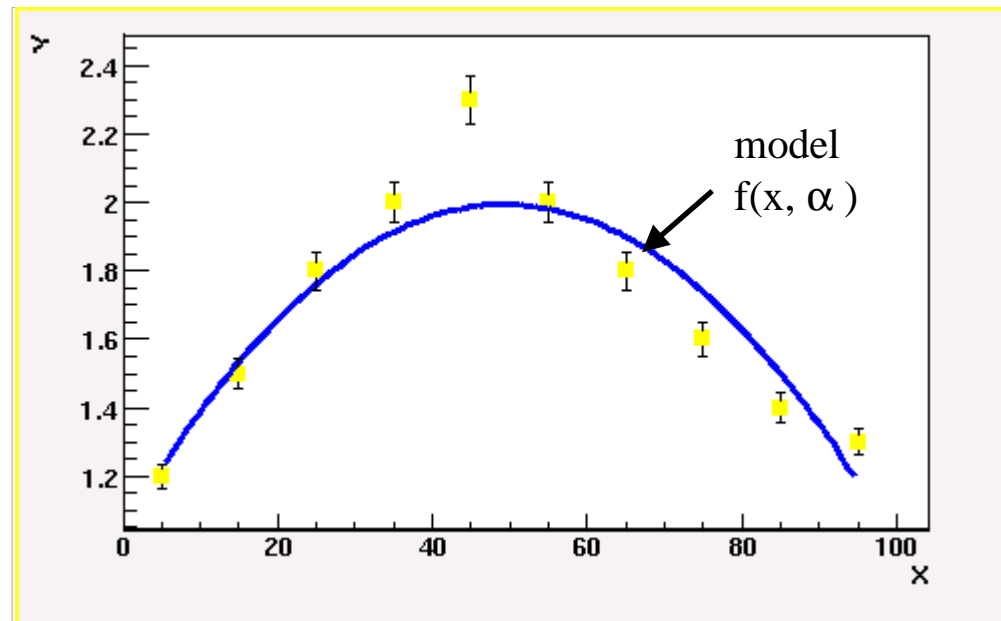
Note that here the bigger α the less χ_{max}^2 is .

Case of indirect measurements : set of n independent measurements y_i at points x_i

$$M = \sum_{i=1}^n \frac{(y_i - f(x_i, \vec{\alpha}))^2}{\sigma_i^2}$$

$f(x_i, \vec{\alpha})$

vector of sought parameters



solution $\vec{\alpha}_s$ is obtained by minimizing M with respect to each parameter

$$\frac{\partial M}{\partial \vec{\alpha}} = 0$$

$\vec{\alpha}_s$ may be tested by a Chi square test

Maximum likelihood method :

$$L(\vec{\alpha}) = \prod_{i=1}^n f(x_i, \vec{\alpha}) \quad x_i : \text{sample of } n \text{ independent values of the same random variable.}$$

$$\ln(L(\vec{\alpha})) = \sum_{i=1}^n \ln(f(x_i, \vec{\alpha}))$$

$$\frac{\partial \{\ln(L(\vec{\alpha}))\}}{\partial \vec{\alpha}} = 0$$

to find the values of the probability density parameters

example : Poisson distribution

$$f(x, \lambda) = \frac{\lambda^x}{x!} e^{-\lambda}$$

$$\frac{\partial}{\partial \lambda} \left(\sum_{i=1}^n \ln \left(\frac{\lambda^{x_i}}{x_i!} e^{-\lambda} \right) \right) = 0$$

$$\frac{\partial}{\partial \lambda} \left(\sum_{i=1}^n (\ln(\lambda^{x_i}) - \ln(x_i!) - \lambda) \right) = 0 \quad \Rightarrow \quad \sum_{i=1}^n \left(\frac{x_i}{\lambda} - 1 \right) = 0 \quad \Rightarrow \quad \lambda = \frac{1}{n} \sum_{i=1}^n x_i = \bar{x}$$

Propagation of errors :

We measure $y(x_i)$, a function of n independent random variables x_i

$$\sigma^2(y) = \sum_{i=1}^n \left(\frac{\partial y}{\partial x_i} \right)_{\vec{x} = \vec{\mu}_x}^2 \sigma^2(x_i)$$

The diagram illustrates the components of the error propagation formula. Arrows point from the following text to the corresponding parts of the equation:

- variance of y** points to $\sigma^2(y)$.
- variance of x_i** points to $\sigma^2(x_i)$.
- square of partial derivative of y with respect to x_i calculated for $\vec{x} = \vec{\mu}_x$** points to the term $\left(\frac{\partial y}{\partial x_i} \right)_{\vec{x} = \vec{\mu}_x}^2$.
- mean values of x_i** points to the subscript $\vec{x} = \vec{\mu}_x$.

Beware that this formula does not work if the random variables are correlated.

Propagation of errors :

We measure r $y_j(x_i)$ functions of n random variables x_i

$$T = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \dots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \dots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_r}{\partial x_1} & \frac{\partial y_r}{\partial x_2} & \dots & \frac{\partial y_r}{\partial x_n} \end{pmatrix}_{\vec{x}=\vec{\mu}_x} \quad \text{evaluated at } \vec{x}=\vec{\mu}_x$$

C_x covariance matrix of x_i

$C_y = T C_x T^T$ Covariance matrix of y_i . Its diagonal terms are the variances of y_i .

Monte-Carlo Techniques

Monte Carlo generation of random numbers and random processes

Named after the renowned Casino quarter of Monaco (on the French Riviera).

Every scientific computing language has a uniform pseudo-random number generating function whose range is $[0, 1]$. The uniform unit random number will thereafter be called x and its probability density $f(x)$:

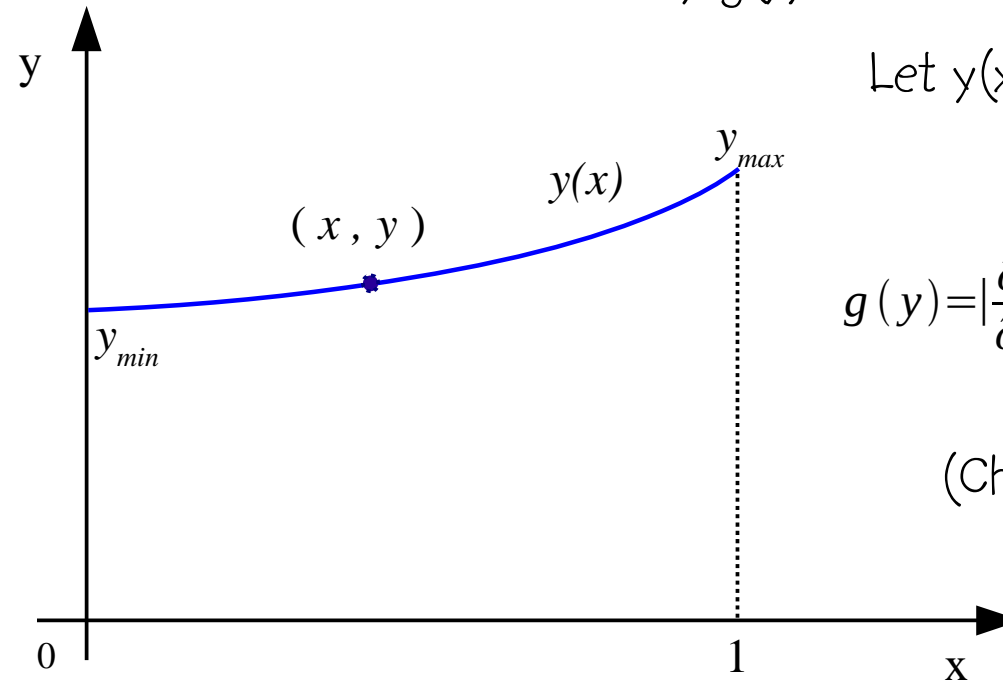
$$f(x)=1 \text{ for } 0 \leq x \leq 1$$

$$f(x)=0 \text{ for all other values}$$

Using x , we can numerically sample random variables according to any probability density. This is heavily utilized in Monte Carlo simulation of subatomic phenomena.

Inverting the distribution function :

y is a random variable with probability density $g(y)$.



Let $y(x)$ be defined.

$$g(y) = \left| \frac{\partial x}{\partial y} \right| f(x) = \frac{\partial x}{\partial y} f(x)$$

(Change of variables)

$$\int_0^x f(x') dx' = \int_{y_{min}}^y g(y') dy' \Rightarrow x = G(y)$$

where $G(y)$ is the distribution function of y .

(since $f(x) = 1$)

then if G can be inverted : $y = G^{-1}(x)$

Sampling x and then applying this relationship results in a random variable y which is distributed according to $g(y)$

example 1 : exponential law (decay of unstable object or free path of interacting particle)

$$g(y) = \lambda e^{-\lambda y} \quad \text{with } 0 \leq y < \infty$$

$$G(y) = \int_0^y \lambda e^{-\lambda y'} dy' = 1 - e^{-\lambda y}$$

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

x being uniformly distributed over $] 0, 1]$

example 2 : normal distribution with mean μ and standard deviation σ .

$$g(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2}$$

Inverting the distribution function is not possible. The technique uses another algorithm called the Box-Muller method.

x and Φ are two uniform random numbers defined as follow :

$$0 < x \leq 1 \quad \text{and} \quad 0 \leq \Phi \leq 2\pi$$

One then computes r according to : $r = \sqrt{\ln\left(\frac{1}{x^2}\right)}$

and we obtain two independent random numbers normally distributed (with zero mean and unit standard deviation) by :

$$y_1 = r \cos \Phi$$

$$y_2 = r \sin \Phi$$

Using : $\sigma y_1 + \mu$, $\sigma y_2 + \mu$ we generate a normal distribution with mean μ and standard deviation σ

Von Neumann Technique :

In case the distribution function is unknown or cannot be easily inverted and if the random variable is defined on a finite range, y can be generated between y_{\min} and y_{\max} by sampling two uniform random numbers u_i and v_i according to :

$$y_{\min} \leq u_i \leq y_{\max} \text{ and } 0 \leq v_i \leq g_{\max} \quad \text{where } g_{\max} \text{ is the maximal value of } g \text{ in the range } y_{\min} \text{ to } y_{\max} .$$

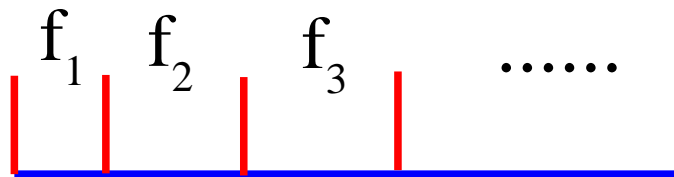
u_i is kept if : $v_i < g(u_i)$

Otherwise, another pair (u_i, v_i) is drawn until the inequality is satisfied.

Discrete variables : Let y be a random discrete variable distributed according to :

$$\sum_{k=1}^n f_k = 1$$

The interval $[0,1]$ can be mapped as follow :



if $x \in f_1$ then $y=0$

if $x \in f_2$ then $y=1$

...

example : Poisson law $f(r) = \frac{\mu^r e^{-\mu}}{r!}$

for $\mu = 1$

$$f(0) = e^{-1} = 0,37$$

$$f(1) = e^{-1} = 0,37$$

$$f(2) = \frac{1}{2} e^{-1} = 0,18$$

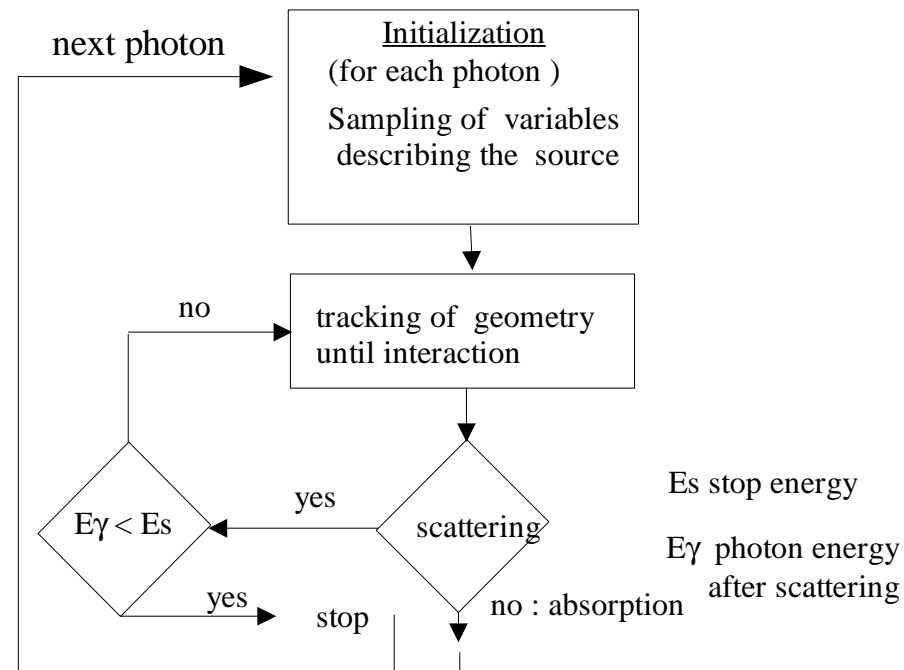
$$f(3) = \frac{1}{6} e^{-1} = 0,06$$



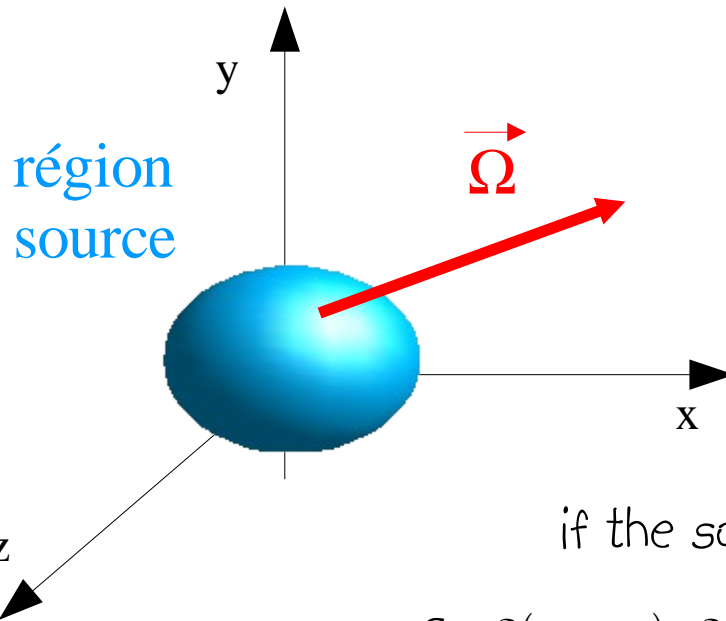
Simulation of stochastic processes :

In subatomic physics all the phenomena are *stochastic*. A Monte Carlo simulation makes use of the numeric sampling techniques we saw to estimate the physical quantities.

Example : simplified simulation of photon interaction in a material.



Source : probability density : $S(x, y, z, \Omega_x, \Omega_y, \Omega_z, E, t)$



E photon energy
 t emission time

If variables are independent :

$$S = S_s(x, y, z) S_\Omega(\vec{\Omega}) S_E(E) S_t(t)$$

if the source is point-like and monoenergetic :

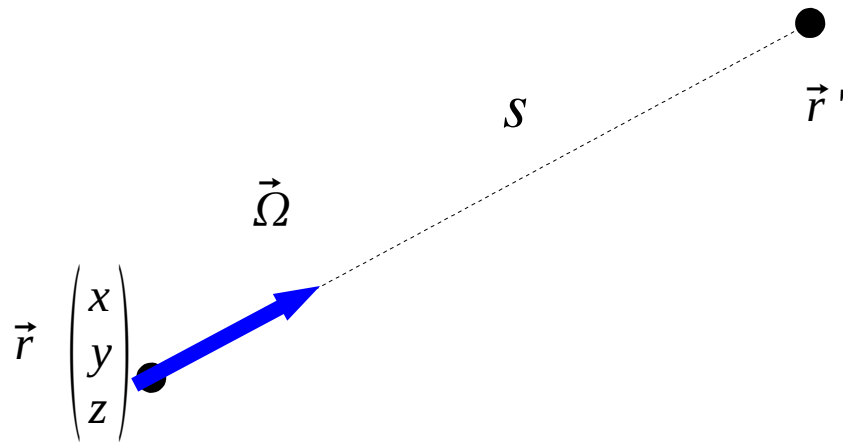
$$S = \delta(x - x_0) \delta(y - y_0) \delta(z - z_0) S_\Omega(\vec{\Omega}) \delta(E - E_0) S_t(t)$$

The initial direction can be computed using :

$$\vec{\Omega} \begin{pmatrix} \Omega_x = \sqrt{1 - \Omega_z^2} \cos \phi \\ \Omega_y = \Omega_x \operatorname{tg} \phi \\ \Omega_z = 1 - 2x \end{pmatrix}$$

where x and ϕ are uniformly distributed between $[0, 1]$ and $[0, 2\pi]$ respectively .

Geometrical tracking and interaction :



Interaction takes place
at : \vec{r}'

free fly between \vec{r} and \vec{r}'

interaction is described by two cross-sections :

σ_a absorption cross-section

σ_s scattering cross-section

total cross-section

$$\sigma_t(E_\gamma) = \sigma_a(E_\gamma) + \sigma_s(E_\gamma)$$

Mean free path of photon given by : $\lambda(E_\gamma) = 1/(\rho \sigma_t(E_\gamma))$

where ρ is the number density of interacting centers (atoms)

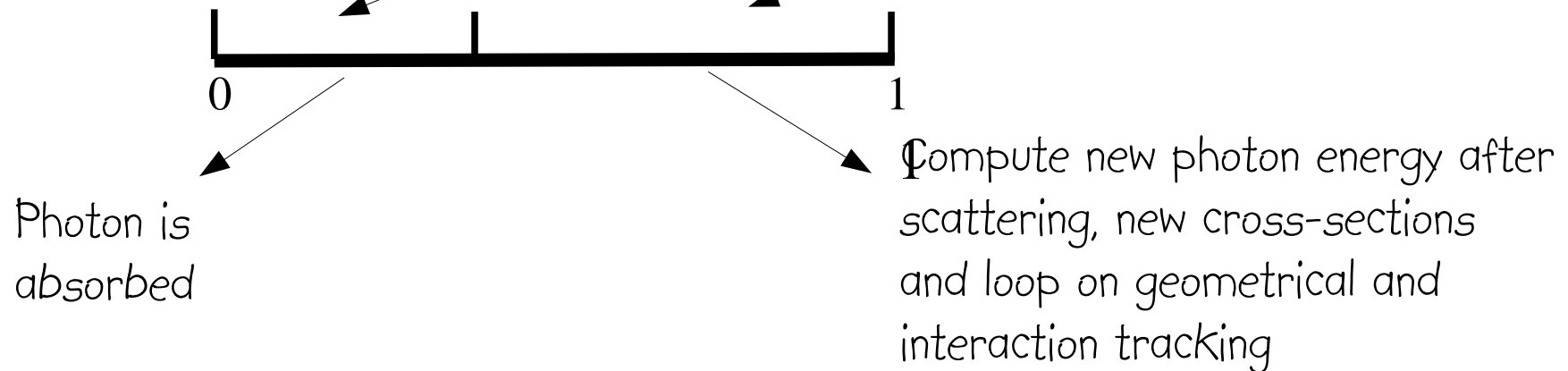
probability density of s :

$$f(s) = \frac{1}{\lambda} e^{-\frac{s}{\lambda}}$$

which can be sampled by : $s = -\lambda \ln(x)$ avec $x \in [0, 1]$

Then new position computed by : $\vec{r}' = \vec{r} + s\vec{\Omega}$

Interaction type is a discrete random variable : $f_a(E_y) = \frac{\sigma_a(E_y)}{\sigma_t(E_y)}$, $f_s(E_y) = \frac{\sigma_s(E_y)}{\sigma_t(E_y)}$



Exemple : tracking of neutrons in water with ROOT

How to install and to use ROOT : <http://root.cern.ch>

Root macros (solutions of the exercices) can be found at <http://lpsc.in2p3.fr/collot>


Each ROOT macro should contain at least a function whose name is the same as the file :
exemple MCNeutron0.C contains `void MCNeutron0(){ }`. See below

```
void MCNeutron0(){ // neutrons in water  
  
    cout << "Welcome to the MC tutorial session" << endl ;  
  
}
```

To execute MCNeutron0.C : `root MCNeutron0.C` ↵

Step 1 : event Loop and pseudo-random unit number generator

```
void MCNeutron1() { // neutrons in water

    gRandom->SetSeed(1800);  set seed of pseudo-random unit number generator

    double range = 5. ; // size of histogram axes

    TCanvas * cl = new Tcanvas("cl","cl",700,700); // ROOT window

    TH2F * hi = new TH2F("rndm2","Neutron tracks",1000,-range,range,1000,-range,range) ;

    double x , y ; // unit vector coordinates
    double s ; // traveled distance
    double phi , phicm ; // lab and CM angles
    double twopi= 2*acos(-1) ; // 2 Pi
    double pi=acos(-1) ; // Pi
    double lambda ; // mean free path
    double T ; // neutron kinetic energy
    double Ti=10. ; // initial neutron kinetic energy in MeV
    double px , py ; // current neutron position
    double pix , piy ; // neutron initial position
```

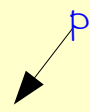
root MCNeutron1.C 

to execute .

Event loop

```
for(Int __t i=0 ; i< 100 ; i++){ // event loop

    px= 0. ; py = 0. ; // initial position
    pix = px ; piy = py ;

    T = Ti ;  pseudo-random unit number

    phi = gRandom->Rndm(1)*twopi ; // initial angle

    x = cos(phi) ; // unit vector coordinates
    y = sin(phi) ;

    px += 2. * x ;
    py += 2. * y ;

    hi->Fill(px,py,1) ;

} // end of event loop

hi->Draw() ; // Draw histogram

}
```

Step 2 : tracking Loop

```
void MCNeutron2(){ // neutrons in water

    gRandom->SetSeed(1800) ;

    double range = 10. ;

    TCanvas * cl = new TCanvas("cl","cl",700,700); // window

    TH2F * hi = new TH2F("rndm2","Neutron tracks",1000,-range,range,1000,-range,range) ; // 2 D histo

    double x , y ; // unit vector coordinates
    double s ; // traveled distance
    double phi , phicm ; // lab and CM angles
    double twopi= 2*acos(-1) ; // 2 Pi
    double pi=acos(-1) ; // Pi
    double lambda ; // mean free path
    double T ; // neutron kinetic energy
    double Ti=10. ; // initial neutron kinetic energy in MeV
    double px , py ; // current neutron position
    double pix , piy ; // last neutron position
    double sc, ssh , sso , stot ; // cross-sections
    double rho = 3.345e22 ; // H2O molecule number density
    int iabs ; // absorption flag

    hi->Draw() ;
```

new variables

modification : draw histogram before event loop

Step 2 : tracking Loop

```
while ( iabs == 0 ) { // while no absorption

    x = cos(phi) ; // unit vector coordinates
    y = sin(phi) ;

    sc = 20.e-24 ; // capture cross-section : 20 b
    ssh = 10.e-24 ; // scattering off H : 10 b
    sso = 5.e-24 ; // scattering off O : 5 b

    stot = sc + ssh + sso ; // total cross section

    lambda = 1./(stot * rho) ; // mean free path in cm

    s=-lambda * log(gRandom->Rndm(l)) ; // traveled distance to next interaction

    pix = px ; // coordinates of last interaction
    piy = py ;

    px += s * x ; // current coordinates after interaction
    py += s * y ;

    lineh = new TLine(pix, piy , px , py ) ; // track line
    lineh->SetLineWidth(0.05) ;
    lineh->Draw() ;
```

```
if( gRandom->Rndm(l) < (sc/stot)){ // absorption ?
    iabs = 1 ;
    cout << "absorption of event " << i << endl ;
}

else { // elastic scattering

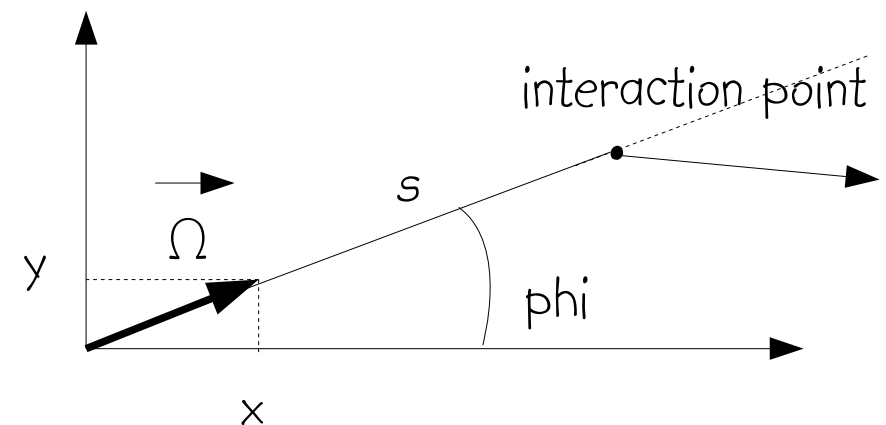
    phi -= pi/4.*(-0.5+gRandom->Rndm(l)) ; // Lab scattering angle

    T /= 10. ; // Kinematic energy divided by 10

    if(T<1.e-9) { T= 1.e-9; } // min thermal neutron energy

}

} // end of tracking loop
```



root MCNeutron2.C ↵ to execute .

Physics

Hypotheses :

Neutron capture cross-section varies as $1/v$.

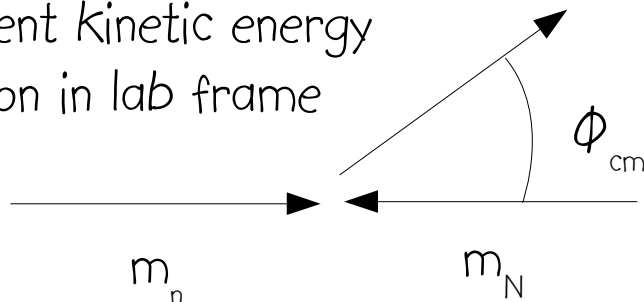
Elastic scattering is isotrope in centre-of-mass frame for H and O.

T : incident kinetic energy of neutron in lab frame

T' : kinetic energy of scattered neutron in lab frame

m_n neutron mass

m_N nucleus mass



Φ_{cm} randomly drawn according to :

$$0 \leq \Phi_{cm} \leq 2\pi$$

$$\Phi_{lab} = \text{atan} \left(\frac{\sin \Phi_{cm}}{(\cos \Phi_{cm} + m_n/m_N)} \right)$$

if $\cos \Phi_{cm} \leq -m_n/m_N$: $\Phi_{lab} = \text{atan} \left(\frac{\sin \Phi_{cm}}{(\cos \Phi_{cm} + m_n/m_N)} \right) + \pi$

$$\frac{T'}{T} = \frac{m_n^2 + m_N^2 + 2 m_n m_N \cos \Phi_{cm}}{(m_n + m_N)^2}$$

exercice : show these expressions

Step 3 : frame for cross-sections and calculation of kinematic variables

Add three cross-section functions :

```
double SigCapture(double T){ // capture cross-section on H , T in MeV
    return 20.e-24 ; // 20 b
}

double SigScatteringO(double T) { // scattering cross-section for Oxygen
    return 10.e-24 ; // 10 b
}

double SigScatteringH(double T) { // scattering cross-section for Hydrogen
    return 5.e-24 ; // 5 b
}
```

In while loop (tracking loop),
call cross-section functions :

```
sc = SigCapture(T) ; // capture cross-section
ssh = SigScatteringH(T) ; // scattering off H
sso = SigScatteringO(T) ; // scattering off O
```

Step 3 : frame for cross-sections and calculation of kinematic variables

add declaration

```
double A,B,C,D ; // final energy parameters
```

Implementation of elastic scattering block

```
else { // elastic scattering

    phicm = gRandom->Rndm(l) * twopi ; // CM scattering angle

    if ( gRandom->Rndm(l) < ssh/(ssh+sso) ) { // hydrogen
        A = 1. ; B = 2. ; C = 2. ; D=4. ;
    }
    else { // oxygen
        A = 1./16. ; B = 257. ; C = 32. ; D=289. ;
    }

    phi -= atan ( sin(phicm) / ( cos(phicm) + A ) ) ; // Lab scattering angle

    if(cos(phicm) < -A) { phi -= pi ;}

    T *=(B+C*cos(phicm))/D ; // lab kinetic energy

    if(T<1.e-9) { T= 1.e-9; } // min thermal neutron energy 1 meV

}
```

root MCNeutron3.C ↵

to execute

Step 4 : cross-sections as a function of T

Capture cross-section : $n+p \rightarrow d+\gamma+2.2 \text{ MeV}$

```
double SigCapture(double T){ // capture cross-section on H , T in MeV

    double v = sqrt( 2. * T / 939.56 ); // neutron speed - beta = v/c
    double r = 100.*0.664e-24 * (7.3e-6 / v ); // sigma varies as 1/v
    return r ;

}
```

Step 4 : cross-sections as a function of T

```
double SigScatteringO(double T) { // scattering cross-section for Oxygen

    double Sig0 ;
    double E0 ;
    double alpha ;
    double r ;
    if ( T < 1.e-7 ) { // T in MeV < 100 meV
        Sig0=12. ; E0=1.e-9 ; alpha = 0.249 ;
    }
    if ( T >= 1.e-7 && T < 0.3 ) { // 100 meV < T < 0.3 MeV
        return 3.8e-24 ;
    }
    if ( ( T >= 0.3 ) && ( T < 4. ) ) { // 0.3 MeV < T < 4 MeV
        Sig0=3.8 ;
        E0=3 ;
        alpha=0.247 ;
    }
    if ( T >= 4. ) { // 4 MeV < T
        Sig0=2. ;
        E0=4. ;
        alpha=0.557 ;
    }

    r = 1.e-24 * Sig0 / pow((T/E0),alpha) ; // power scaling low
    return r ;
}
```

Elastic scattering off oxygen.

Described as a series of power laws :

$$\sigma(T) = \sigma(T_0) (T/T_0)^{-\alpha}$$

except for a few cases where it could be approximated as a constant.

Step 4 : cross-sections as a function of T

Elastic scattering off hydrogen.

root MCNeutron4.C ↵

to execute

```
double SigScatteringH(double T) { // scattering cross-section for Hydrogen
double Sig0 ;
double E0 ;
double alpha ;
double r ;
if ( T < 1.e-7) { // T < 100 meV
  Sig0=90. ;
  E0=2e-9 ;
  alpha=0.379. ;
}
if ( T >= 1.e-7 && T < 1.e-3) { // 100 meV < T < 1 keV
  return 2*20.4e-24 ;
}
if ( (T >= 1.e-3) && (T < 0.1) ) { // 1 keV < T < 0.1 MeV
  Sig0=20.4 ;
  E0=0.001 ;
  alpha=0.0978 ;
}
if ( (T >= 0.1) && (T < 1.) ) { // 0.1 MeV < T < 1. MeV
  Sig0=13. ;
  E0=0.1 ;
  alpha=0.491 ;
}
if(T >= 1.) { // 1 MeV < T
  Sig0=4.2 ;
  E0=1. ;
  alpha=0.69 ;
}
r = 2.e-24 * Sig0 / pow((T/E0),alpha) ; // power scaling low
return r ;
}
```

Step 5 : final step , how to color tracks

Energy color code : Red energetic , Blue slow neutrons

Color code declaration :

```
Color__t lc[7] = {kRed,kOrange,kYellow,kGreen,kCyan,kBlue,kMagenta}; // line colors
```

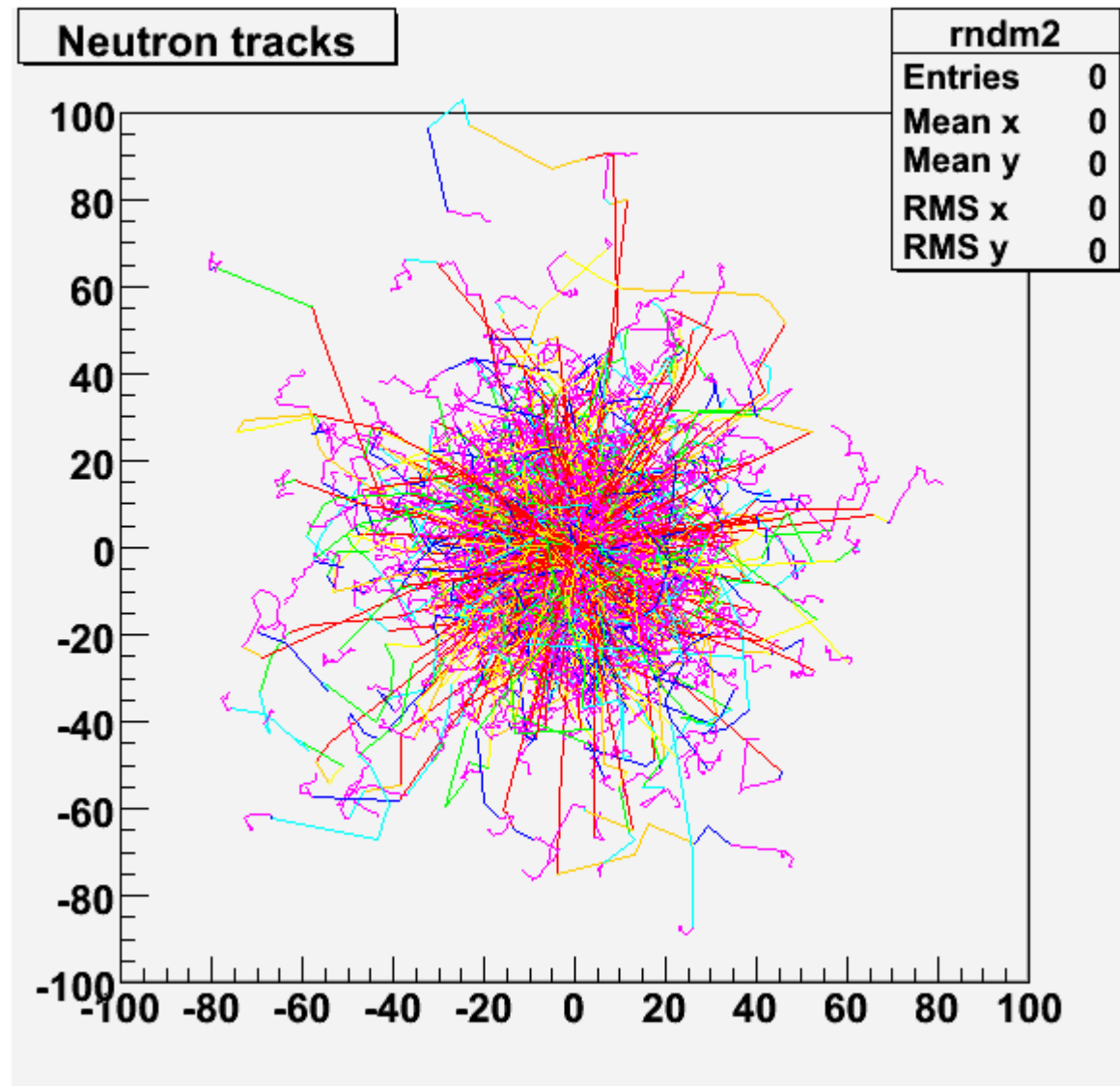
Assignment of color to tracks :

```
lineh = new TLine(pix, piy , px , py ) ; // track line  
lineh->SetLineWidth(0.05) ;  
int k = (int) ((1.-T/Ti)*7.) ; // track line color choice  
if(k>6) k=6 ;  
lineh->SetLineColor(lc[k]) ;  
lineh->Draw() ;
```

root MCNeutron.C ↵

to execute

What you should get : 1000 simulated neutrons propagating in water



To learn more :

- Data Analysis, Sigmund Brandt , Springer
- Statistics for nuclear and particle physicists, Louis Lyons, Cambridge University Press
- Statistical Methods in experimental physics : W.T. Eadie, D. Drijard, F.E. James, R. Roos, B. Sadoulet , North-Holland
- Monté Carlo Methods : M. Kalos & P. Whitlock, John Wiley & Sons (1986)