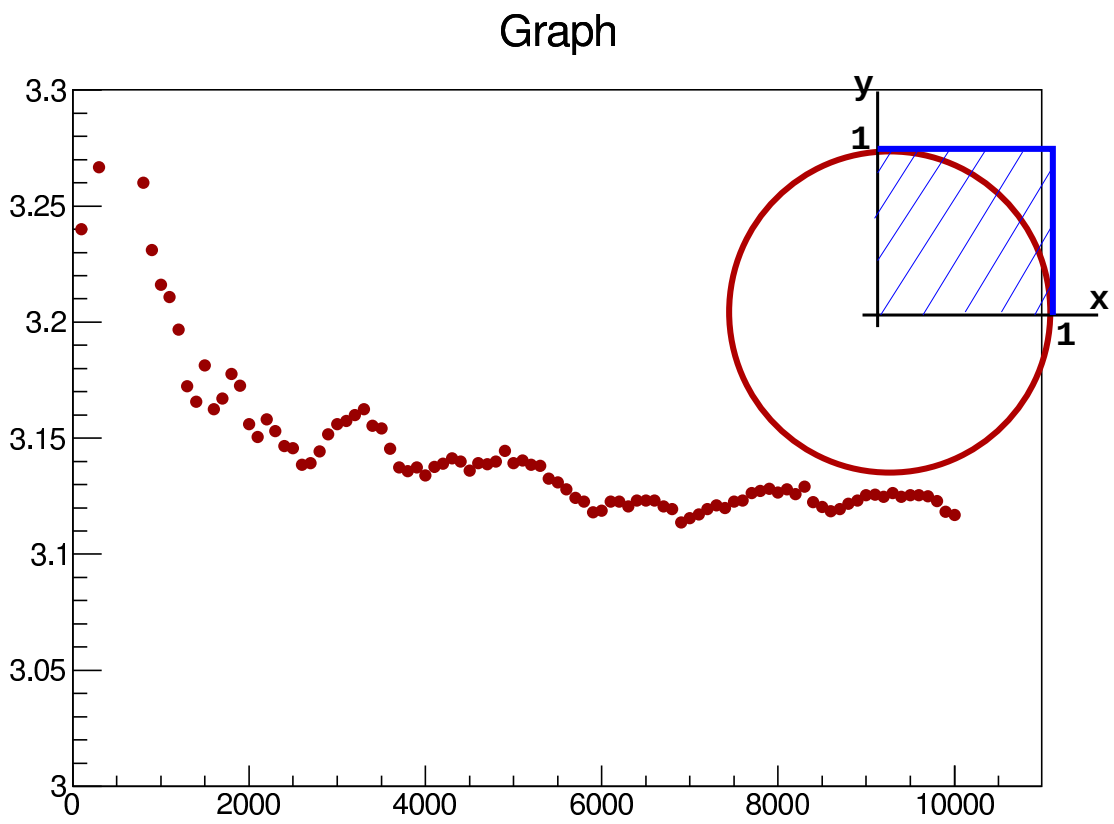
Pi evaluation



Monte Carlo integration

✔ we want to evaluate the following integral:

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

✔ remember that the expectation value of the function $f(x)$ for x distributed according to a PDF $p(x)$

$$\langle f \rangle = \int_a^b f(x)\,p(x)\,dx \qquad \text{with:} \int_a^b p(x)\,dx = 1$$

✔ choosing x to be uniformly distributed in the interval $[a,b]$, one has:

$$\begin{aligned} p(x) &= \frac{1}{b-a} \\ \langle f \rangle &= \int_a^b f(x)\,p(x)\,dx = \frac{1}{b-a}\int_a^b f(x)\,dx \end{aligned}$$

MC integration

$$\begin{aligned} F &= \int_a^b f(x)\,dx \\ &= (b-a)\,\langle f \rangle \\ &= \frac{(b-a)}{N}\sum_{i=1}^{N} f(x_i) \end{aligned}$$

x_i is a random variable uniformly distributed in the interval $[a,b]$

error estimation

$$\begin{aligned} \sigma_F &= (b-a)\sigma_{\langle f \rangle} \\ \sigma_f^2 &= \left\langle f^2 \right\rangle - \langle f \rangle^2 \\ \sigma_{\langle f \rangle}^2 &= \frac{\sigma_f^2}{N} \end{aligned}$$

$$\sigma_F = (b-a)\frac{\sigma_f}{\sqrt{N}} = (b-a)\frac{1}{\sqrt{N}}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(f(x_i)\right)^2 - \left(\frac{1}{N}\sum_{i=1}^{N} f(x_i)\right)^2}$$

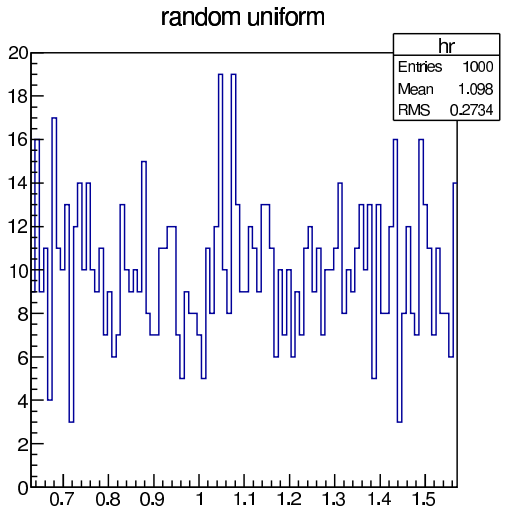
MC integration (cont.)

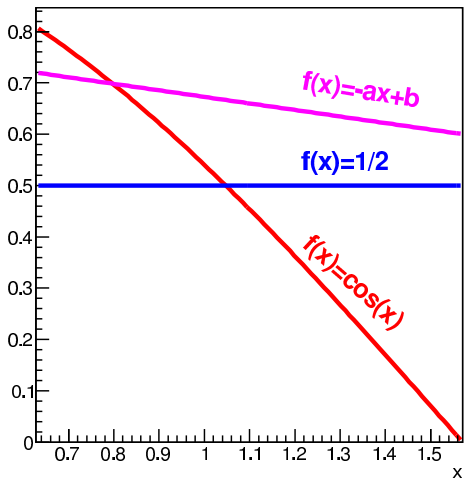
Let's compute the integrals of the functions:

$$\int_{0.2\pi}^{0.5\pi} \frac{dx}{2} = 0.471239$$
$$\int_{0.2\pi}^{0.5\pi} cos(x)\, dx = 0.412215$$
$$\int_{0.2\pi}^{0.5\pi} (ax+b)\, dx = 0.622035$$

Throwing 100 random variable uniformly distributed we obtain the following results:

$$\int_{0.2\pi}^{0.5\pi} \frac{dx}{2} = 0.471239 \pm 0.000000$$
$$\int_{0.2\pi}^{0.5\pi} cos(x)\, dx = 0.413671 \pm 0.007098$$
$$\int_{0.2\pi}^{0.5\pi} (ax+b)\, dx = 0.622280 \pm 0.001037$$





MC integration (cont.)

algorithm

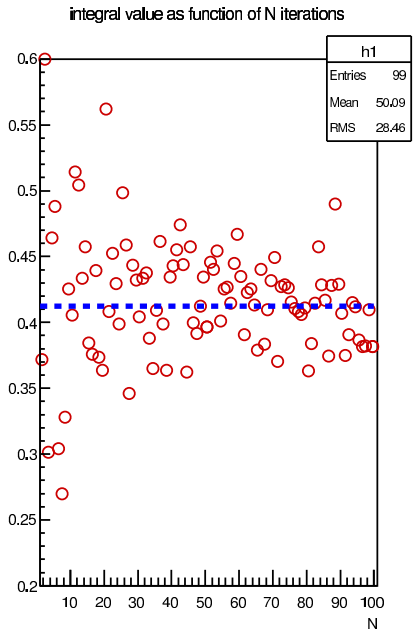
```
double xmin=TMath::Pi()*0.2;
double xmax=TMath::Pi()*0.5;
int N = 1000;

TF1 *f1 = new TF1(''f1'',''TMath::Abs(cos(x))'',xmin,xmax);
TF1 *f2 = new TF1(''f2'',''0.5'',xmin,xmax);
TF1 *f3 = new TF1(''f3'',''-0.4/TMath::Pi()*x+0.8'',xmin,xmax);

...
for (int i=0; i<N; i++) {
   double x = xmin + (xmax-xmin)*gRandom->Uniform();
   double func1 = f1->Eval(x);
   double func2 = f2->Eval(x);
   double func3 = f3->Eval(x);
   F1 += func1;
   F2 += func2;
   F3 += func3;
   f1s += func1*func1;
   f2s += func2*func2;
   f3s += func3*func3;
 }
double f1m = F1/N; //mean
double f2m = F2/N;
double f3m = F3/N;
```

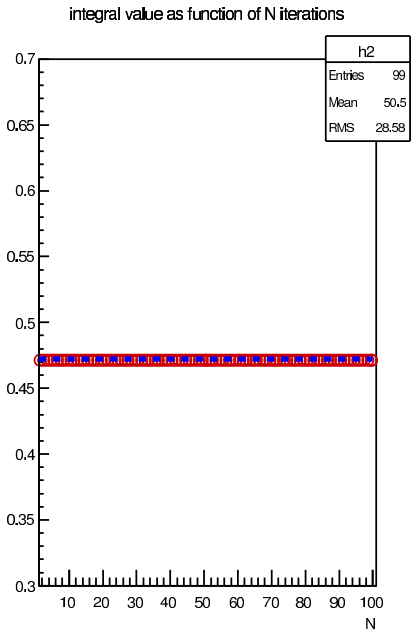
algorithm

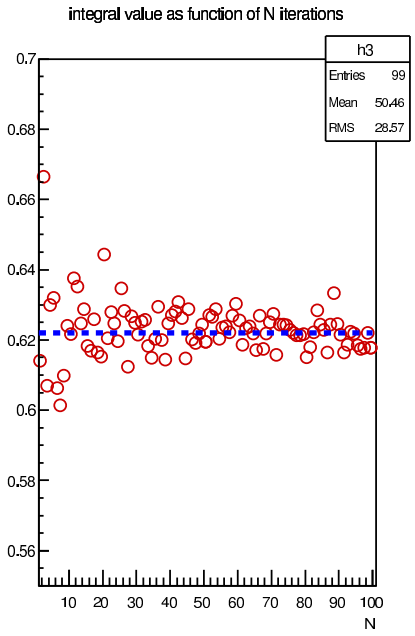
```
// integrals
double I1 = f1m*(xmax-xmin);
double I2 = f2m*(xmax-xmin);
double I3 = f3m*(xmax-xmin);

// variances
double Var1 = f1s/N - f1m*f1m;
double Var2 = f2s/N - f2m*f2m;
double Var3 = f3s/N - f3m*f3m;

// errors
double E1 = (xmax-xmin)/sqrt(N)*sqrt(Var1);
double E2 = (xmax-xmin)/sqrt(N)*sqrt(Var2);
double E3 = (xmax-xmin)/sqrt(N)*sqrt(Var3);
```

MC integration (cont.)

Let's check the integral value as function of the number of random variables generated N







$$F = \int_{0.2\pi}^{0.5\pi} cos(x)\, dx$$

$$F = \int_{0.2\pi}^{0.5\pi} \frac{dx}{2}$$

$$F = \int_{0.2\pi}^{0.5\pi} (ax + b)\, dx$$

Empty Slide

Reduction variance techniques

- ✔ The $cos(x)$ function varies much more in the interval of integration that the others
- ✔ Its integral value evaluation presents the largest variance. Why?
- ☞ Because we are sampling uniformly and the regions close to zero where the function is more important are sampled with the same importance as others where the function is smaller!
- ☞ In the framework of the **importance sampling technique** an additional pdf $p(x)$ can be used to rend the integrand smooth!

Importance sampling

- ✔ Rend smooth our integrand by applying a pdf $p(x)$

$$F = \int_a^b f(x)\,dx = \int_a^b \frac{f(x)}{p(x)}\,p(x)\,dx$$

- ✔ If the pdf is normalized in the integral interval $[a, b]$

$$\int_a^b p(x)\,dx = 1$$

and x is a variable distributed according to $p(x)$, then

$$\left\langle \frac{f}{p} \right\rangle = \int_a^b \frac{f(x)}{p(x)}\,p(x)\,dx$$

- ✔ Let's make a variable change

$$\int_a^b \frac{f(x)}{p(x)}\,\underbrace{p(x)\,dx}_{p(y)dy}$$

$p(x)dx = p(y)dy$
if y is distributed uniformly in $[0, 1]$ then
$\int_0^1 p(y)dy = 1 \Rightarrow p(y) = 1$

The transformation between x and Y can be obtained by:

$$\int_a^x p(x')dx' = \int_0^y dy' \quad \Rightarrow \quad \boxed{y = \int_a^x p(x')dx'}$$

Importance sampling (cont.)

✔ From the transformation of variables we have a relation between x and y

$$y = \int_a^x p(x')dx' \quad \Rightarrow \quad x(y)$$

Generating a random variable y uniformly between $[0, 1]$ and applying the transformation relation $x(y)$ we get random variables x distributed according to $p(x)$

$$F = \int_a^b f(x)\,dx = \int_a^b \frac{f(x)}{p(x)}\,p(x)\,dx = \int_0^1 \frac{f\,[x(y)]}{p\,[x(y)]}\,dy = \left\langle \frac{f}{p} \right\rangle_y = \frac{1}{N}\sum_{i=1}^{N} \frac{f\,[x(y_i)]}{p\,[x(y_i)]}$$

✔ Exercise: make the following integral

$$\int_{0.2\pi}^{0.5\pi} cos(x)\,dx$$

expected = 0.412215

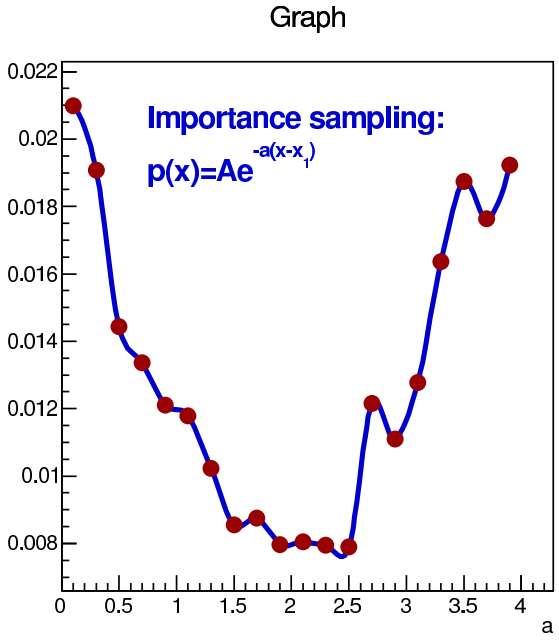
MC = 0.432225 +/- 0.025083 (100 deviates generated)

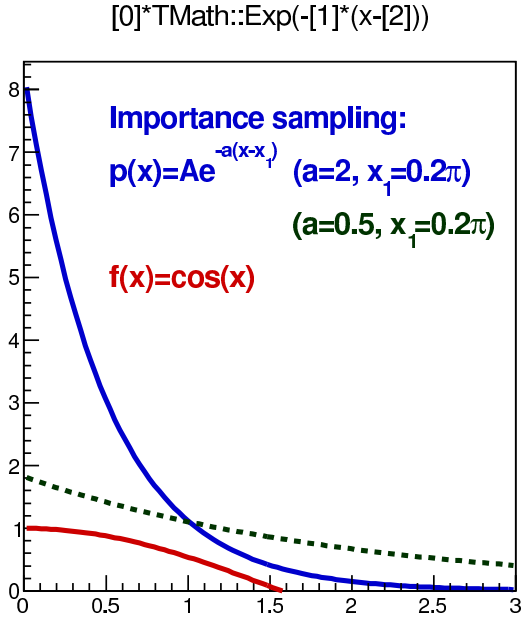
What about using importance sampling with a pdf: $p(x) \propto e^{-ax}$?

Importance sampling (cont.)

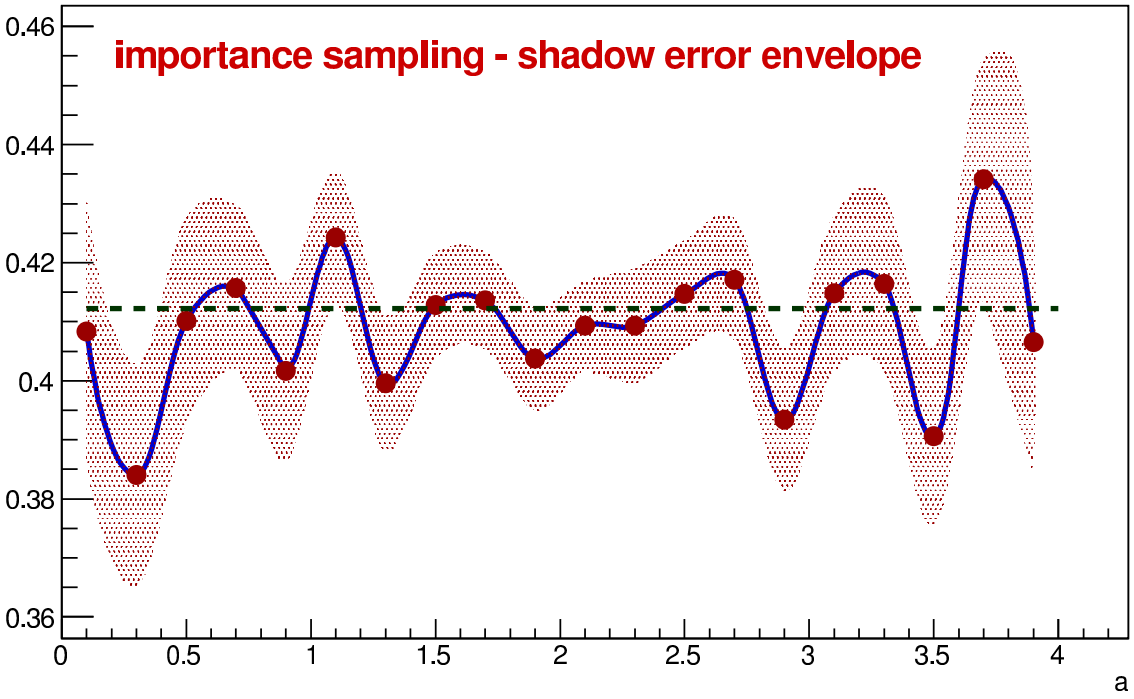
The function PDF shape matters?

Let's study the variation of the integral error with the a parameter of the exponential





Importance sampling (cont.)



Simulation

- **Simulation** is very important for understanding real situations or for modelling the behaviour of a system

  it is largely used on particle and astroparticle physics for designing the instruments used for particles detection

- the various real conditions the system has can be introduced easily in a simulated process

- Suppose you had to design a detector system for detecting photons coming from Compton scattering on a material?

  I assume my gamma source emits a beam very colimated along an axis (x for instance) and in between I have a block of material where Compton is going to happen...

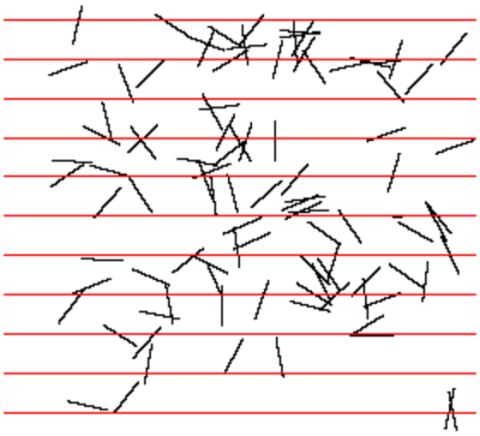
  What we need to know?

Buffon's needle problem

Buffon's needle problem is a question first posed in the 18th century by Georges-Louis Leclerc where simulation can help us a lot!

A needle of length $\mathbf{l}$ is thrown randomly onto a grid of parallel lines, separated by a distance $\mathbf{d}$, with $\mathbf{d} > \mathbf{l}$

What is the probability that a needle intersects a line?



Buffon's needle problem

The probability of crossing the line it's the ratio between the needle "perpendicular distance",
$\frac{l}{2}\sin\theta$
and the lines separation distance, $\mathbf{d}$

$$P(\theta) = \frac{\frac{l}{2}\sin\theta}{d}$$

All θ are likely, so we need to calculate an average probability for θ range:
$2 \times [0, \pi]$

Generating random variables

- The common problem is to have a variable x distributed according a given distribution function $p(x)$
  we are going to make a change of variable such that the number of events (randoms) generated is independent of the used variable

  $$dN = p(x)dx = p(y)dy$$

- Suppose that y is a random variable distributed in $[0,1]$ and x starts at x_0

  $$p(y) = 1 \quad \Rightarrow \quad \int_0^y dy' = \int_{x_0}^x p(x')dx' \quad \Rightarrow \quad y = \int_{x_0}^x p(x')dx'$$

- For generating a random variable x in a interval $[x_0, x_1]$ we just make sure that:

  $$\int_{x_0}^{x_1} p(x)dx = 1$$

  and we invert the relation above (boxed) giving us $x(y)$.
  Provided a random y in $[0,1]$ we generate a random x in $[x_0, x_1]$.

uniform [0,1] → uniform [a,b]

- Let's transform a variable
  y uniformly distributed in $[0,1]$
  into a variable
  x uniformly distributed in $[a,b]$

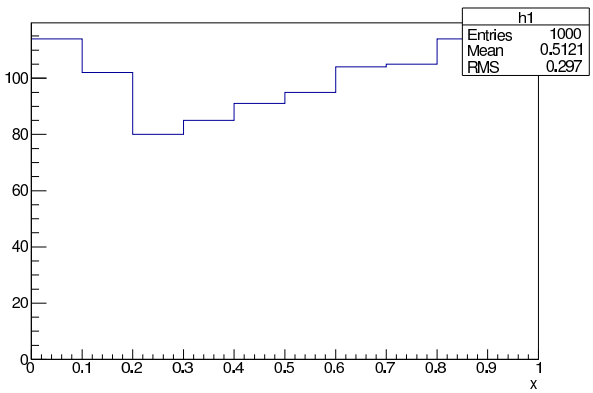
- Normalization of $p(x)$:

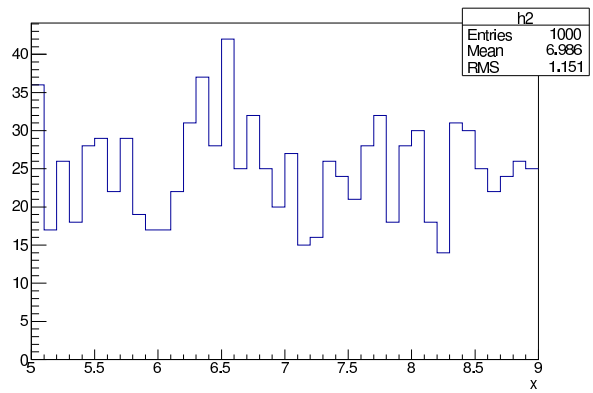
  $$p(x) = \frac{1}{b-a}$$

- Transformation:

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

  $$x = a + (b-a)\, y$$





uniform [0,1] → exponential $[0, \infty]$

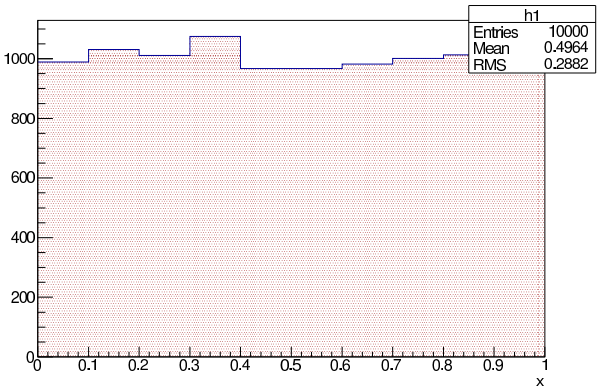
- Let's transform a variable
  - ☞ y uniformly distributed in $[0, 1]$ into a variable
  - ☞ x distributed according to $p(x) \propto e^{-x}$ in $[0, \infty]$
- Normalization of $p(x)$:

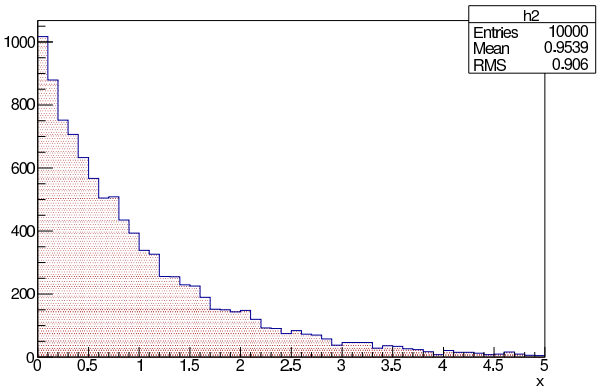
$$k \int_0^{+\infty} e^{-x}\, dx = 1$$
$$\Rightarrow k \left[-e^{-x}\right]_0^{\infty} = 1 \Rightarrow \quad k = 1$$
$$p(x) = e^{-x}$$

- Transformation:

$$y = \int_0^x e^{-x'} dx' = \left[-e^{-x'}\right]_0^x = 1 - e^{-x}$$

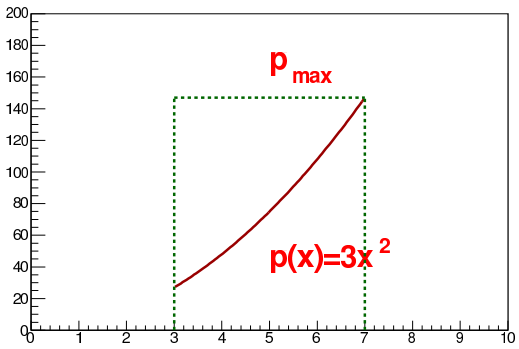
$$x = -\ln(1 - y)$$

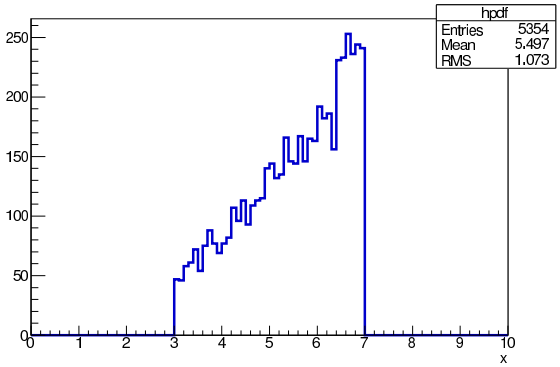




randoms gen: acceptance-rejection

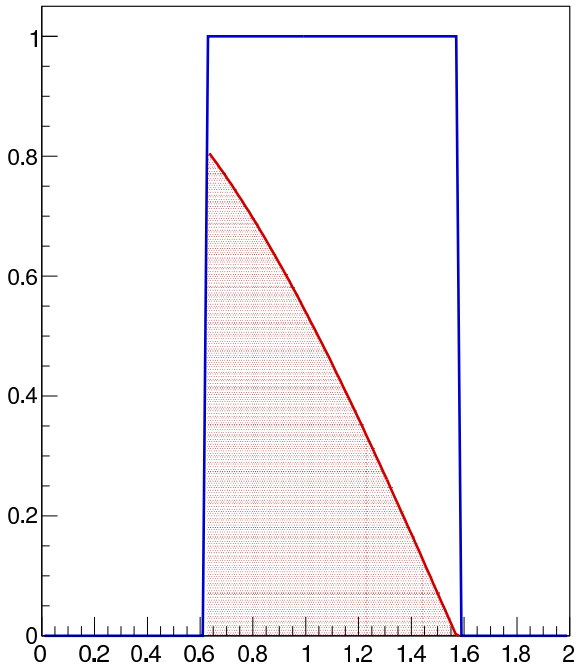
- For applying the method before said we need to integrate the pdf and invert it
- In case it is not possible we can use the more generic but less efficient method of acceptance-rejection
- For generating a x variable in the interval $[a, b]$ distributed according to a pdf $p(x)$ wo do the following:
  - ☞ generate a uniform random x_R between $[a, b]$
  - ☞ compute $p(x_R)$ and the ratio $\frac{p(x_R)}{p_{max}}$
  - ☞ generate a second random u_R from $U(0, 1)$
  - ☞ if $u_R \leq \frac{p(x_R)}{p_{max}}$ accept the variable x_R





MC integration: acc-rej

- ✔ method introduced by von Neumann
- ✔ for integrating the function we define an envelope with an area
  $A = (x_{max} - x_{min}) * f_{max}$
- ✔ generate two random variables
  - ☞ x_R in in range $[x_{min}, x_{max}]$
  - ☞ f_R in in range $[0, f_{max}]$
- ✔ count the number of events N_R that $f_R \leq f(x_R)$
- ✔ the integral $I = (x_{max} - x_{min})\ f_{max}\ \frac{N_R}{N}$
- ✔ the integral error
  $\sigma_I = \frac{(x_{max}-x_{min})\ f_{max}}{N} \sqrt{N_R\left(1 - \frac{N_R}{N}\right)}$



Using 100 randoms:
$\int_{0.2\pi}^{0.5\pi} cos(x)\ dx = 0.435 \pm 0.037$

random gen: $f(x) = 3x^2$

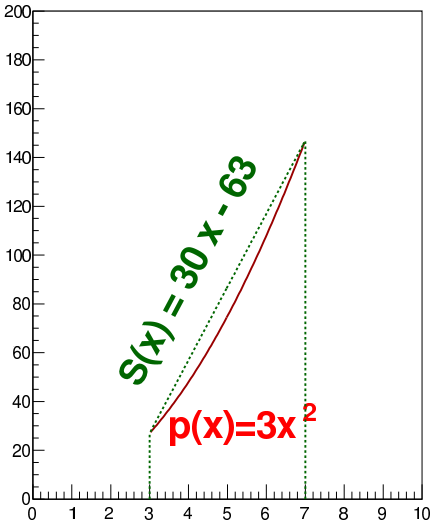
- ✔ The acceptance-rejection method is inefficient if the function varies quickly
- ✔ The fraction of randoms accepted (**efficiency**) is given by:

$$\varepsilon = \frac{\int_a^b f(x)\ dx}{(b-a)\ f_{max}}$$

- ✔ The efficiency can be improved using an auxiliar function $S(x)$ that has a shape close to the one we want to sample

$$\varepsilon_S = \frac{\int_a^b f(x)\ dx}{\int_a^b q(x)\ dx}$$

where $q(x) = C\ S(x) \geq f(x)$ for x in $[a, b]$



efficiencies:

$\varepsilon = \frac{N_R}{N} \simeq 0.535$

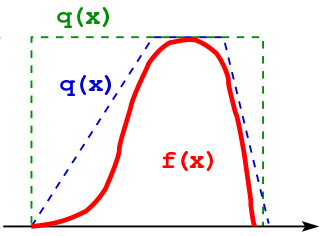
$\varepsilon_S = \frac{N_R}{N} \simeq 0.91$

acc-rej with aux function

For generating a random variable x distributed according to $f(x)$ in the interval $[a, b]$

1. Find auxiliar function $S(x)$ with a shape close to the function $f(x)$ we want to sample
   - ☞ integrable, invertible
   - ☞ from $S(x)$ we define a pdf $p(x)$ and we apply the transformation $y \in U[0,1] \rightarrow x \in U(p(x))$
   - ☞ invert transformation equation: $x(y)$
   - ☞ using $x(y)$ and generating y uniformly between $[0, 1]$ we obtain the random variable x_R distributed according to $p(x)$
2. Define a function $q(x) = C\, S(x)$ such that $q(x) \geq f(x)$ in the interval $[a, b]$
3. Generate random variable x_R according to $p(x)$ and compute $\frac{f(x_R)}{q(x_R)}$
4. Generate random $u_R \in U[0,1]$ and accept x_R if:

$$u_R \leq \frac{f(x_R)}{q(x_R)}$$



random gen: $f(x) = 3x^2$

✔ We want to generate a variable according to a function

$$f(x) = 3x^2$$

1. Define the auxiliar function $S(x)$ to improve acceptance-rejection efficiency

$$S(x) = 30x - 63$$

generate random in [3,7] interval according to auxiliar function

get pdf(x): $S(x) \rightarrow p(x)$ with $\int_3^7 p(x)dx = 1$

$k \int_3^7 S(x)\, dx = 1 \quad \Rightarrow \quad k = \frac{1}{348}$

$$p(x) = \frac{1}{348}(30x - 63)$$

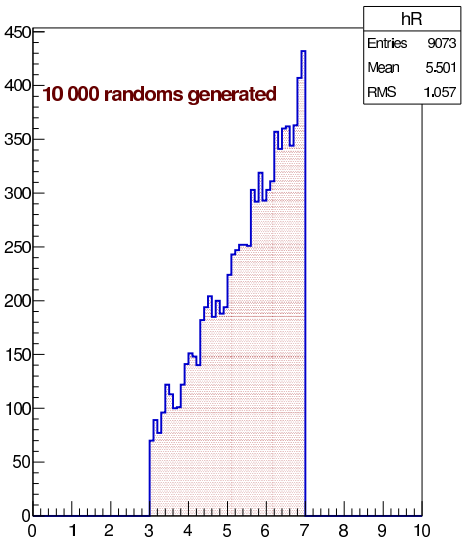
make transformation: $y[0,1] \rightarrow x$ according to $p(x)$

$y = \int_3^x p(x')\, dx' = \int_3^x \frac{1}{348}(30x' - 63)\, dx'$

Derive $x(y)$ from: $15x^2 - 63x + 54 - 348y = 0$

2. define $q(x)$ which in this case is $= S(x)$ and generate x_R according to $p(x)$

3.,4. generate $u_R \in u[0,1]$ and accept x_R if $u_R \leq \frac{f(x_R)}{q(x_R)}$



C++ classes

```
class Func1D {
public:
 Func1D(TF1 *fp=NULL);
 // other constructors?
 ~Func1D();
 void Draw();
 double Evaluate();
 (...)
protected:
 TF1 *p; //integrand function
};
```

```
class IntegratorMC: public Func1D {
public:
 Integrator(double fx0, double fx1, TF1 *fp=NULL) :
            x0(fx0), x1(fx1), Func1D(fp) {;}
 ~Integrator();

 // set function
 void SetIntegrandFunction(TF1*);

 // simple integration
 void IntegralMC(double xmin, double xmax, int N,
                 double& result, double& error);

  // importance sampling
  void IntegralMCIS(double xmin, double xmax, int N,
                    double& result, double& error, TF1* pdf);

  // acceptance-rejection
  void IntegralMCIS(double xmin, double xmax, int N,
                    double& result, double& error, TF1* pdf);
  // other methods
  (...)
protected:
 double x0, x1; // integrand limits
};
```


Computational Physics

Universe

a particle accelerator

Fernando Barao, Phys Department IST (Lisbon)