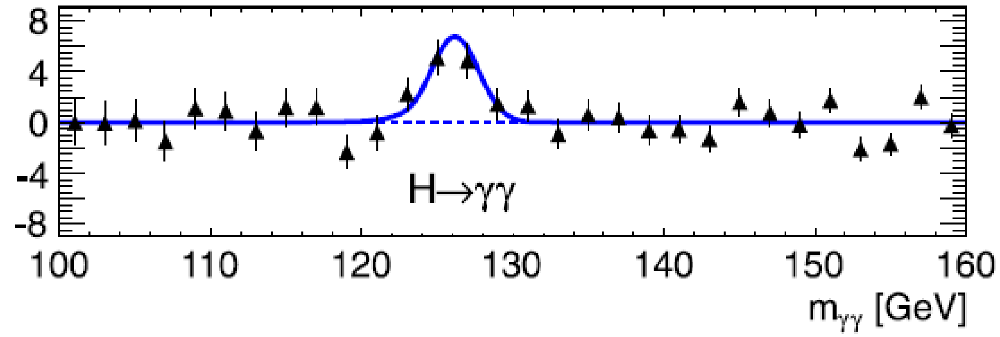
Introduction to
Statistical Data Analysis

Chapter 4

Monte Carlo

Methods



Jan 2024

Introduction

The deterministic systems are described by some mathematical rule. But some systems are not deterministic known as random or stochastic. **The Monte Carlo Method** is a numerical technique for calculating probabilities and related quantities by using sequences of random numbers.

Pioneers

Some Applications

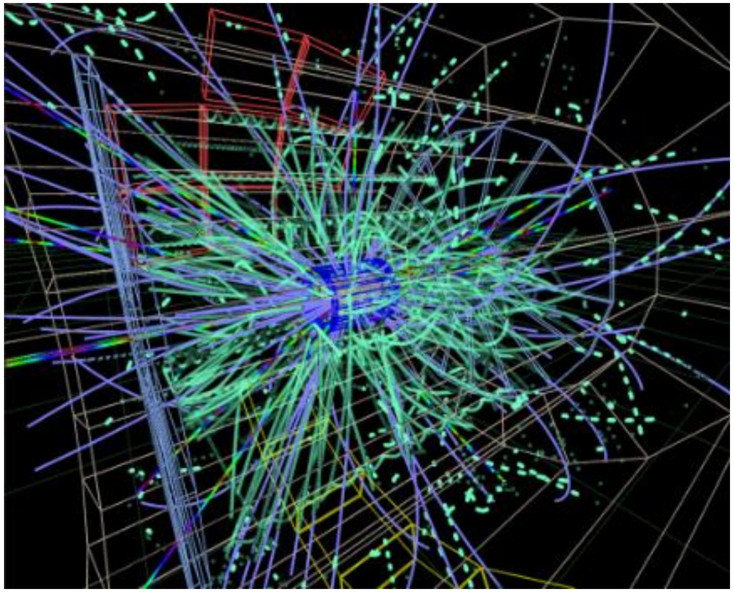
MC method is often used to simulate experimental data.

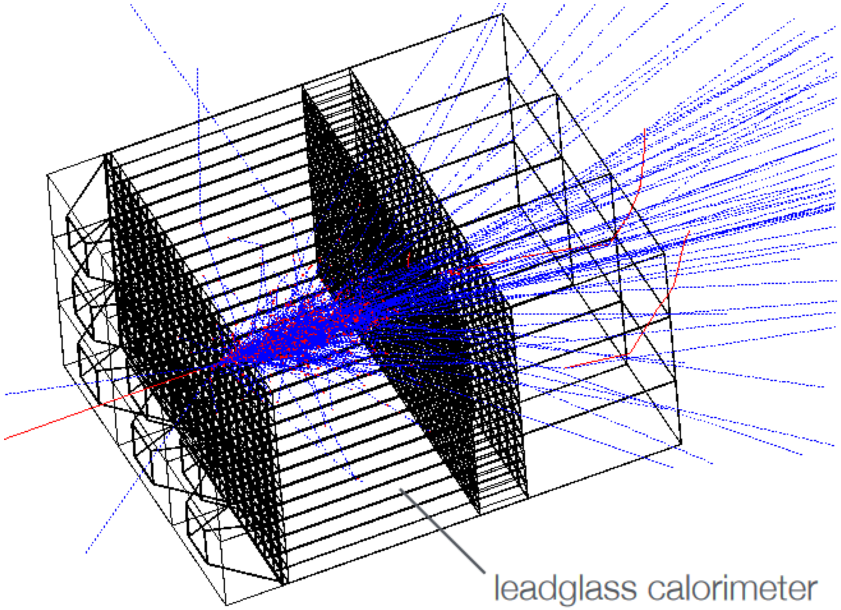
Applications Field	Example
Physics	Simulation of Radioactive decay
Engineering	Tolerance Analysis
Bioinformatics	Protein Folding
Statistics	Distribution Functions
Economy	Modelling Stock Exchange
Medicine	Treatment Planning in Radiation Therapy

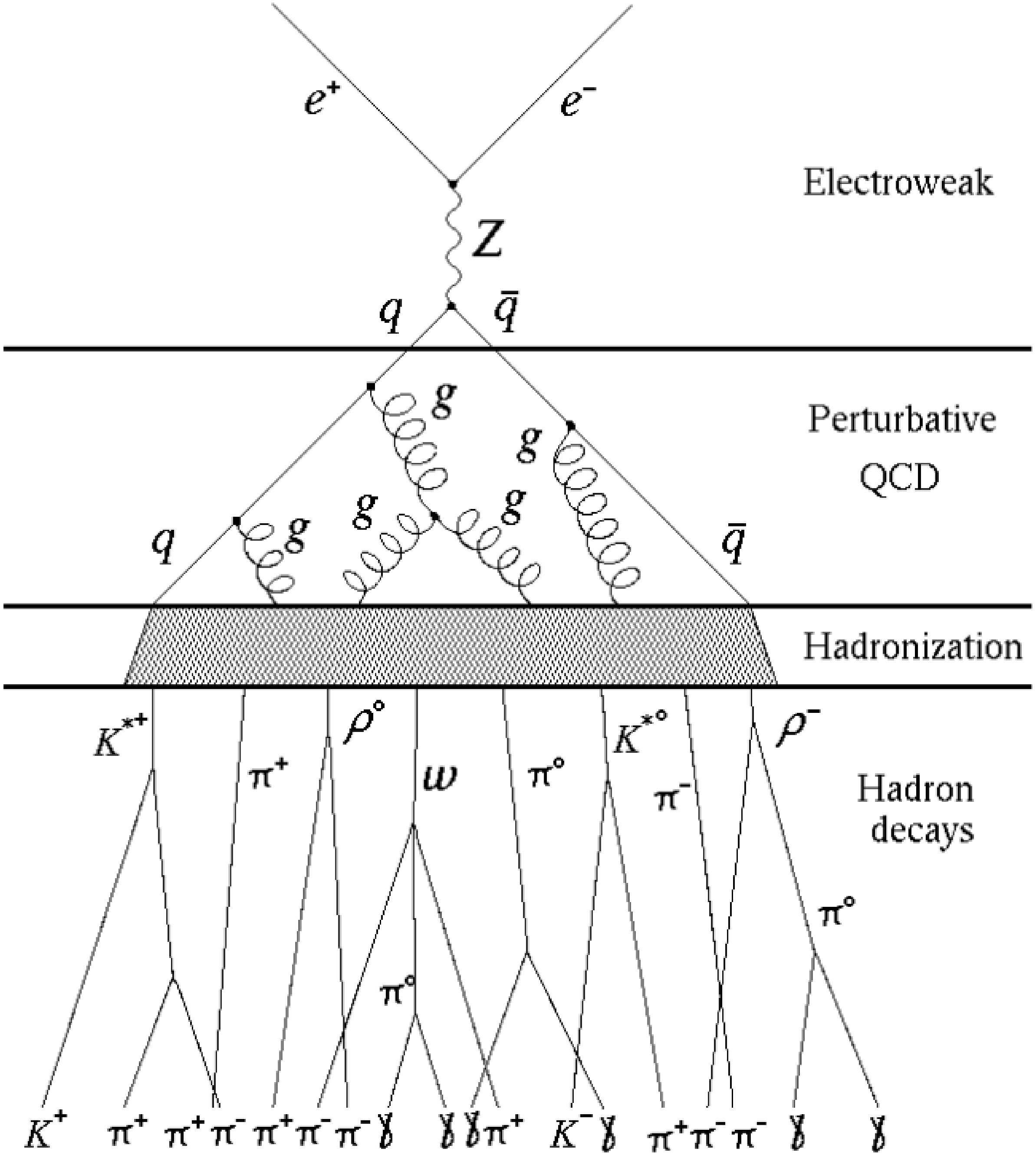
In Particle Physics

Simulation of experimental data is typically done in two stages:

1. Event generation (Pythia, Sherpa, Herwig, …)
2. Detector simulation (GEANT4)





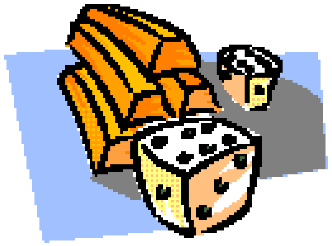


e+
e-
Electroweak
Z
q
q̄
g
g
Perturbative
QCD
q
g
g
g
q̄
Hadronization
K*+
ρ°
ρ-
π+
ω
π°
K*°
π-
Hadron
decays
π°
π°
K+ π+ π+ π- π+ π- π- γ γ γ π+ K- γ π+ π- π- γ γ

Random Numbers

In principle, the best way to obtain a series of random numbers:

$$\{x_1 , x_2 , x_3 , \dots x_n\}$$



is to use some process in nature:

- Throw a coin or dice
- Lotto results of last Sunday.
- Number of particles from radioactive decay in every 1 min.
- Number of cosmic muons falling on 10cm x 20 cm area in 10 s.
- Brightness level of a star observed in atmoshpere in every 1 s.

These are not very efficient.

Therefore, **random number generator** algorithms have been developed to be used in computers.

Uniformly Distributed Random Numbers

Linear Conguential Method (1948)

uses 32-bit integers have a period of at most $2^{31} \sim 10^9$.

The method is employed by an equation of the form:

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

where mod means modulo. Constants *a*, *b* and *m* are chosen carefully such that the sequence of numbers becomes chaotic and evenly distributed. The resulting values are therefore more correctly called **pseudorandom**.

RULES:

- First initial number, x_0, called seed, is selected.
- $m > x_0,\ a,\ b \geq 0$
- The range of values is 0 to *m.* The period generator is *m-1*.
- Divide by *m* to convert to 0. to 1.

Example 1

if we select $a = b = x_0 = 7$ and m = 10 results in the squence:

$$x = \{7, 6, 9, 0, \quad 7, 6, 9, 0, \; ...\}$$

and dividing each number by 10 gives:

$$r = \{0.7,0.6,0.9,0.0,\ 0.7,0.6,0.9,0.0,\ ...\}$$

very poor!

Some good generators are proposed:

- IBM in 1960: $x_{i+1} = (\ 69069\ x_i\ )\ \mathrm{mod}\ 2^{31}\text{-}1$
- Park and Miller: $x_{i+1} = (\ 16807\ x_i\ )\ \mathrm{mod}\ 2^{31}\text{-}1$
- P.L. L'Ecuyer: $x_{i+1} = (\ 40692\ x_i\ )\ \mathrm{mod}\ 2147483399$

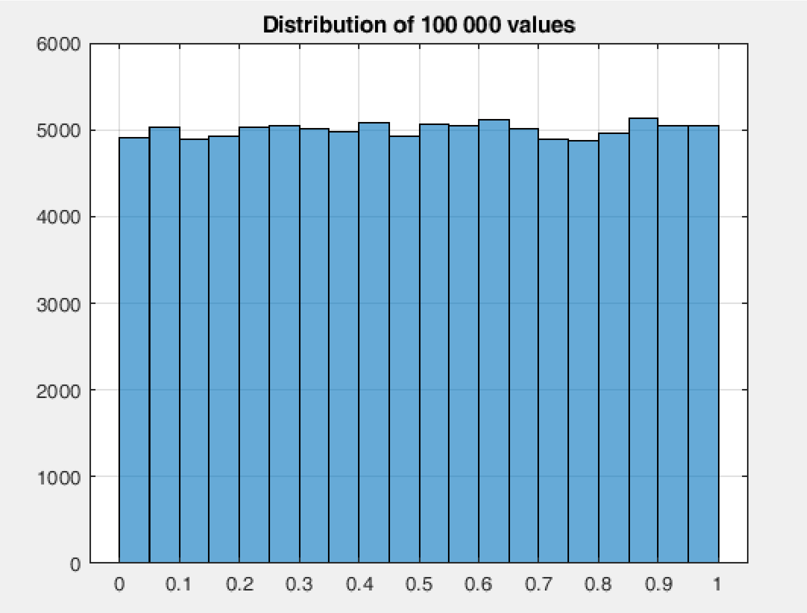
Some advanced algorithms:

- RANLUX has period of 10^{143} (Root uses this algorithm)
- Mersenne twister has period of 10^{6000}

Example 2: Implementation of RANDU

```
m = 2^31-1; % maximum value
x = 314;    % the seed

for i = 1:10
    x = mod(69069*x, m); % integer
    r = x / m;           % real
    disp(r)
end
```



<u>OUTPUT</u>

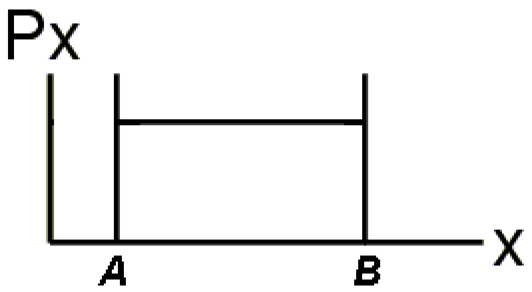
```
0.0101
0.5352
0.6915
0.3512
0.5424
0.8728
0.6736
0.2940
0.4968
0.9160
```

Uniformly Distributed Random Numbers

Assume that ***r*** is a uniform random real number in the range [0,1]

A and *B* are real numbers,

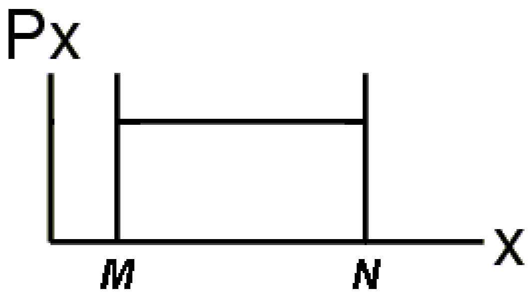
M and *N* are integer numbers, then the value



$$x = A + (B-A)r$$

will be *random real number* in the range [A, B]

$$x = M + int(Nr)$$



will be *random integer number* in the range [M,N].

Two examples:

Random decay angle in the range [0, 2π]

```
phi = 2*pi*r;
```

Random integer in the range [1, 6]

```
D = 1 + int(6*r);
```

Built-in Random Number Generators

C++

https://cplusplus.com/reference/random/

Obsolete C++

`int k = rand();` returns random integer in the range [0, RAND_MAX]

`double r = double(k)/RAND_MAX;`

Root

```
TRandom rnd;
```

`double = rnd.Uniform();` returns random real in the range (0,1).

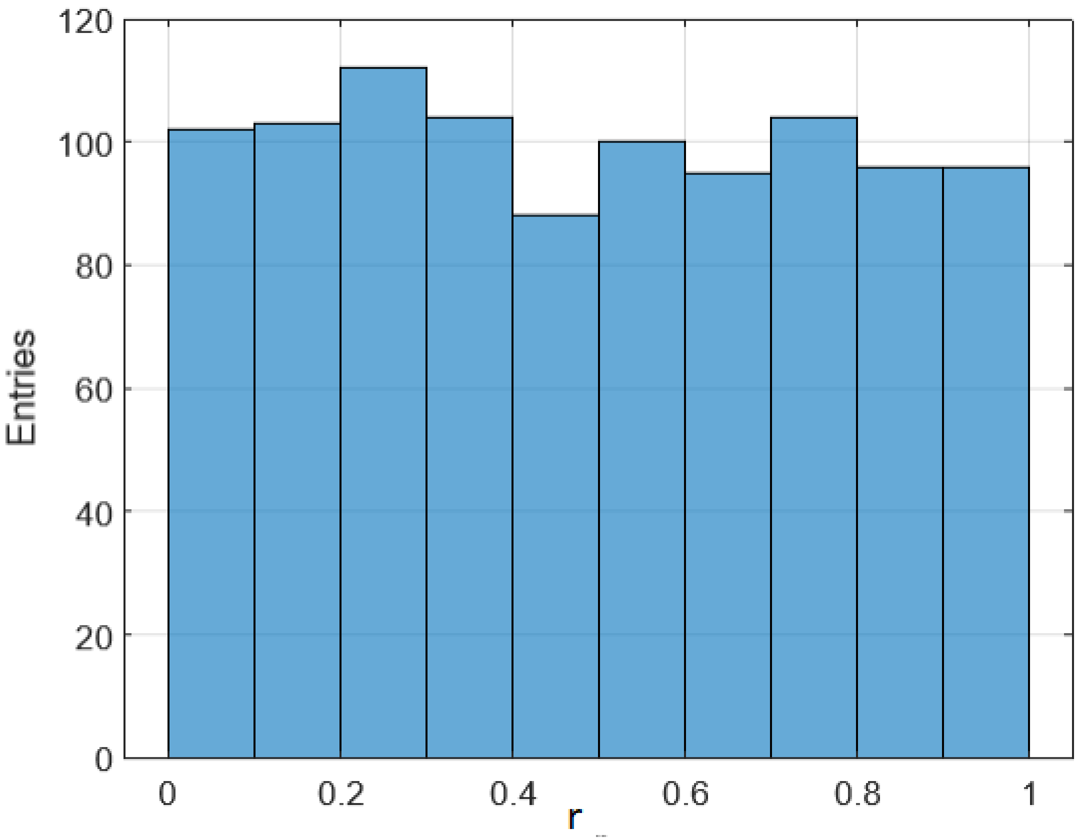
MATLAB

`r = rand` returns random real in the range (0,1).

`r = rand(3,2)` generates random 3x2 random matrix.

Example 3: Using rand function

```
n = 1000;
r = rand(n,1);
histogram(r,10)
xlabel('r')
ylabel('Entries')
grid on
```



Box-Muller Algorithm for S.N.D.

1. Generate two uniformly distributed random numbers u_1 and u_2 in the range [0,1]

2. Set

$$\phi = 2\pi u_1, \qquad r = \sqrt{-2 \ln u_2}$$

3. Then

$$z_1 = r \cos \phi \quad \text{and} \quad z_2 = r \sin \phi$$

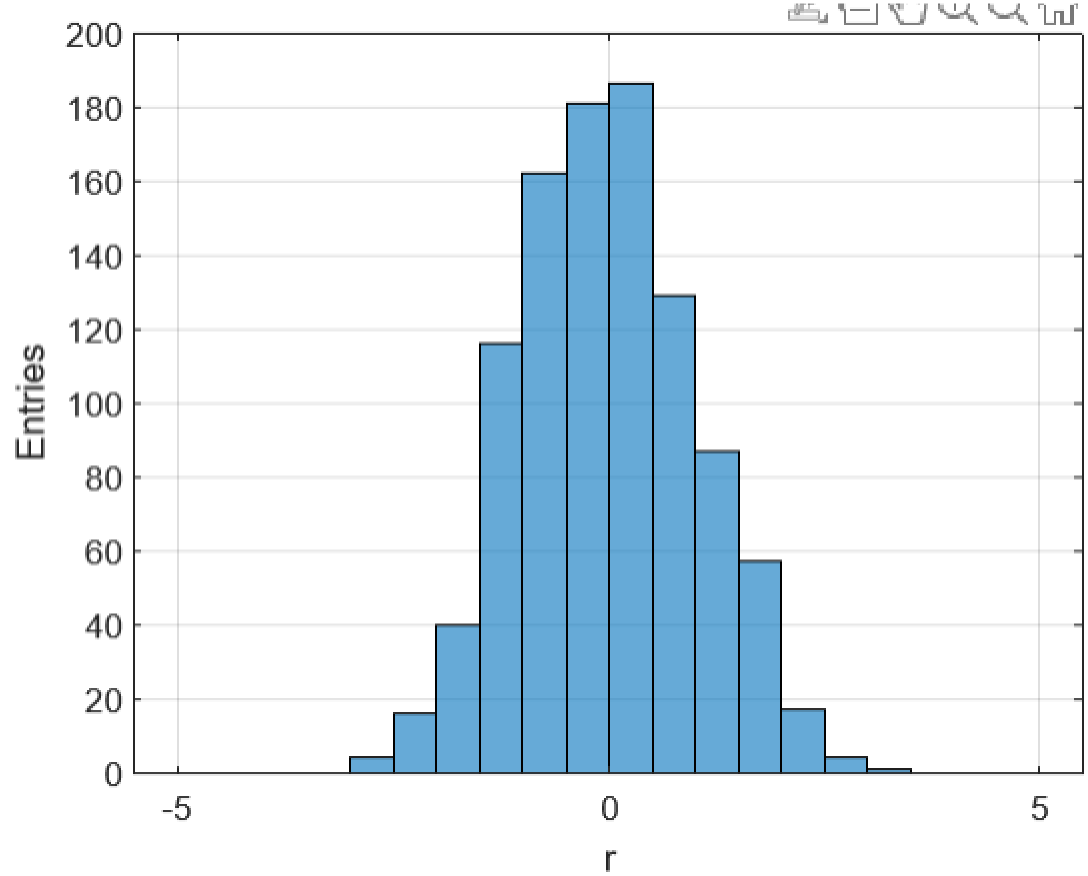
are two independent rv's following a standard normal distribution

To get a random number taken from standard normal distribution:

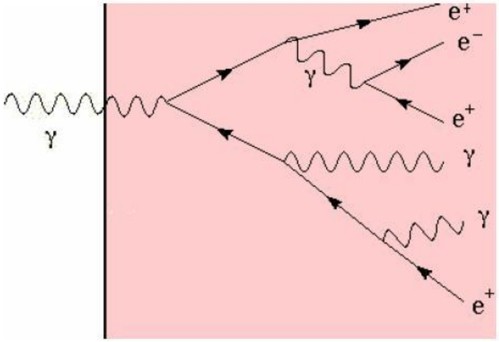
- use function **`normrnd(0,1)`** in MATLAB.
- use function **`rnd.Gaus(0,1)`** in Root.

Example 4: using normrnd function

```
n = 1000;
r = normrnd(0,1, n,1);
histogram(r,12)
xlabel('r')
ylabel('Entries')
grid
```



Example 5: ECAL Detector Resolution



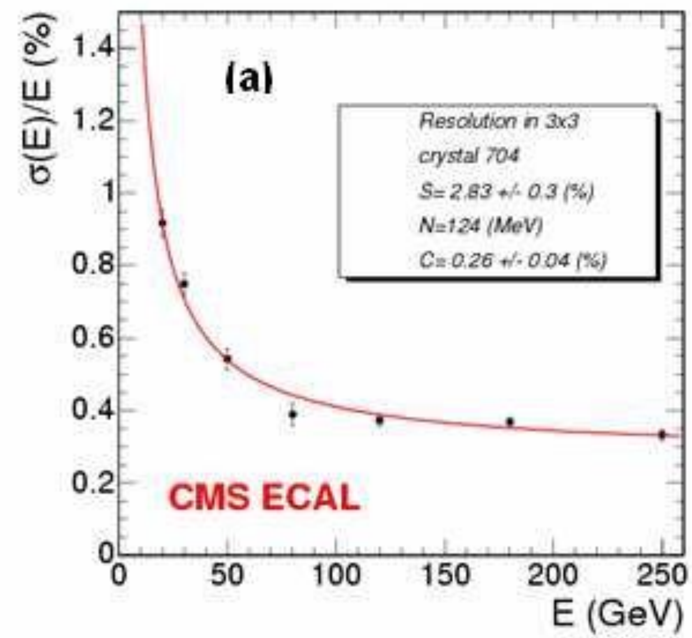
Photon energy measured, *E*, is proportional to number of electrons, *n*, which are counted in the ECAL shower, because the number of charges is related with energy deposition in any space. Thus: $E \propto n$.
But, this counting is a statistical process carrying a statistical uncertainty $\sigma_E \propto \sqrt{n}$. Then,

$$\frac{\sigma_E}{E} \propto \frac{\sqrt{n}}{n} = \frac{1}{\sqrt{n}}$$

or

$$\frac{\sigma_E}{E} = \frac{R}{\sqrt{n}}$$

for ATLAS *R* = 0.10 GeV$^{1/2}$
for CMS *R* = 0.05 GeV$^{1/2}$

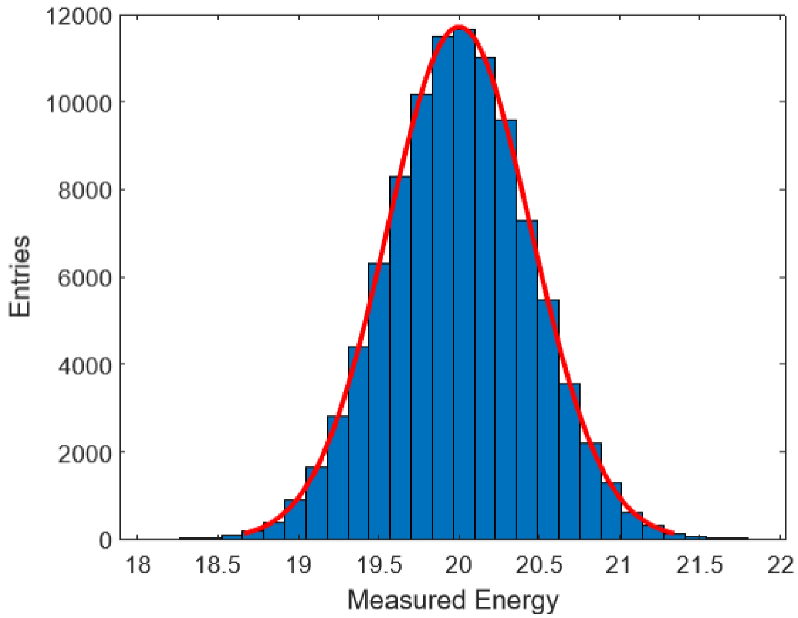


We can model ECAL by smearing the photon energies (*E* is in GeV).

$$\sigma_E = 0.1\sqrt{E}$$

For example, assume that E = 20 GeV. Then, the following code block can roughly simulate the response of ATLAS ECAL for *n* = 100 000 photons.

```
E = 20;
sigmaE = 0.1*sqrt(E);
n = 1e5;
energy = normrnd(E,sigmaE,n,1);
histfit(energy,30)
xlabel('Measured Energy')
ylabel('Entries')
```

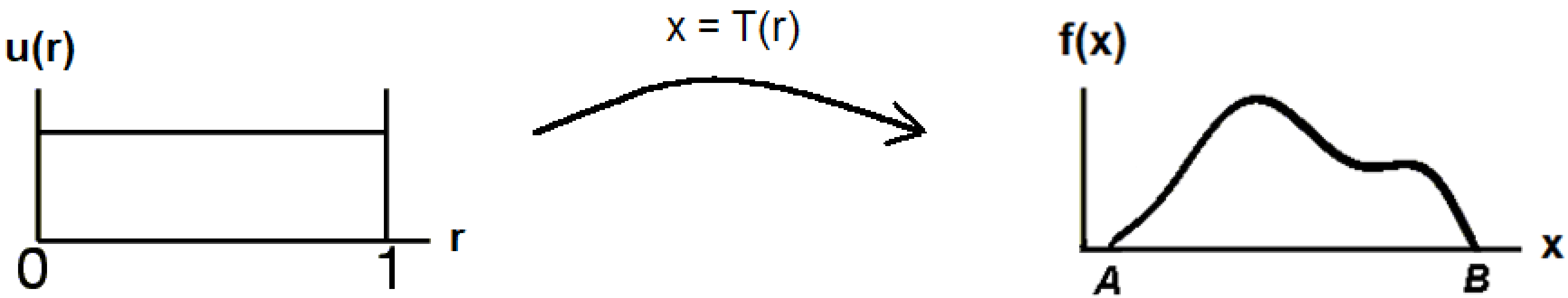


Random Distributions

In simulations of random processes, we often require a non-uniform distribution of random numbers. Two standard methods are:

- The Transformation Method
- The Rejection Method

The aim of both these methods is to convert a uniform distribution of random numbers of the form $u(r)$ into a non-uniform distribution of the form $f(x)$. The values of x can then be treated as simulated measurements



Transformation Method (Continues RV)

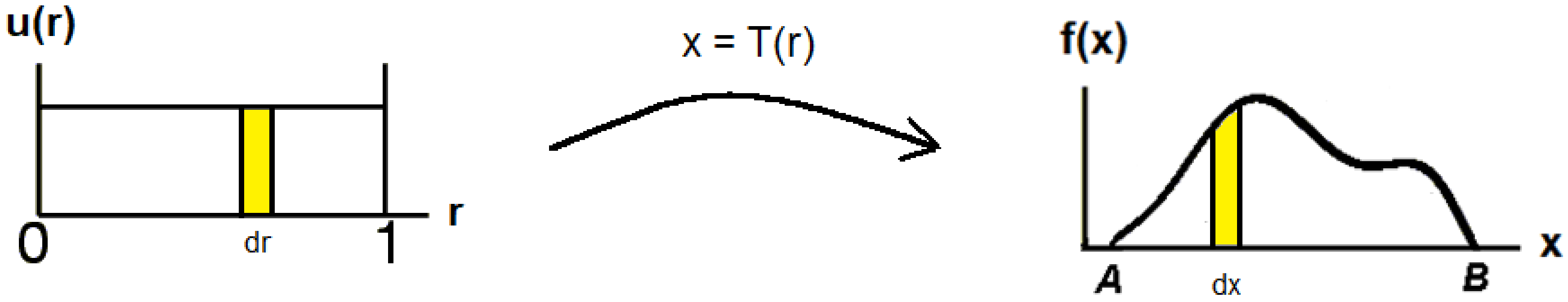
Let *u*(r) be uniform distribution in the range [0,1].

Consider a distribution f(*x*) from which we want to draw random numbers, *x*.

Conservation of random numbers!

$$u(r)dr = f(x)dx$$

The aim is to find a transformation function *x* = T(*r*) such that the distribution of random variable *x* is f(*x*).



Since $u(r) = 1$, we need to solve D.E.

$$dr = f(x)dx$$
$$r = \int f(x)dx$$

Right hand side is CDF:

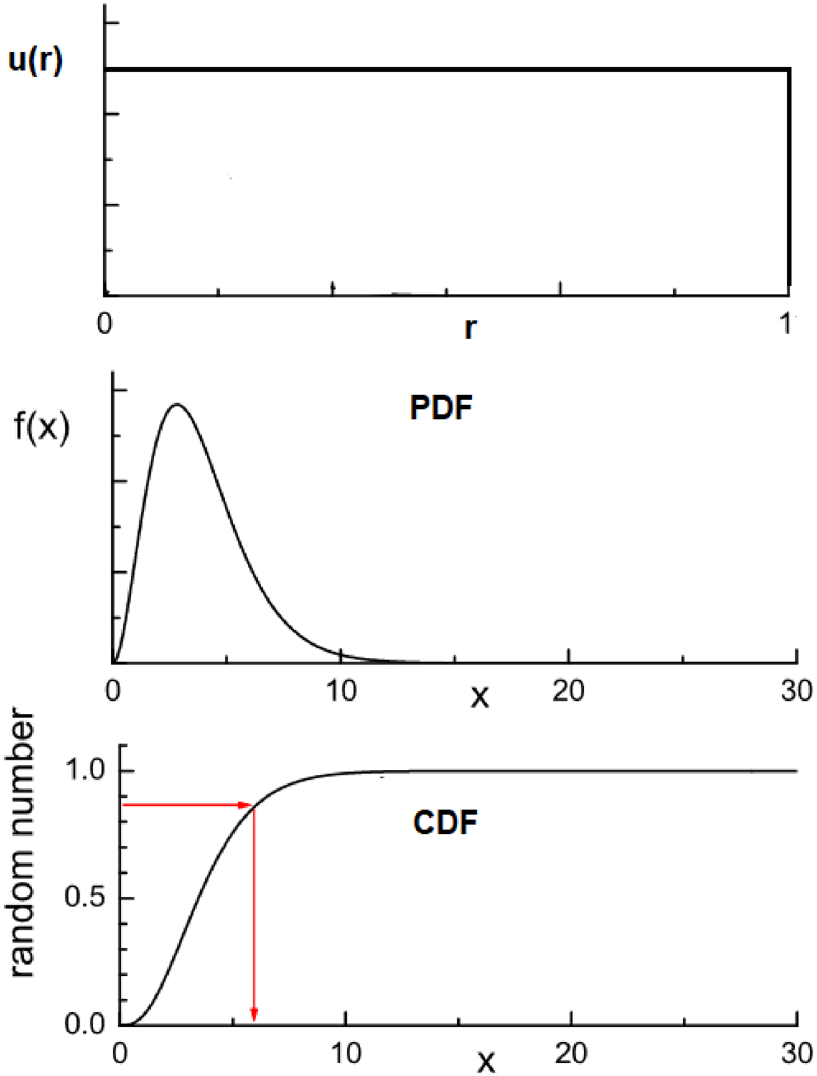
$$F(x) = \int_{-\infty}^{x} f(x)dx = \int_{0}^{r} u(r)dr = r$$

$$F(x) = r$$

We want to get x from inverse CDF.

$$x = F^{-1}(r) = T(r)$$

This function is known as the
Transformation Function.



Example 6

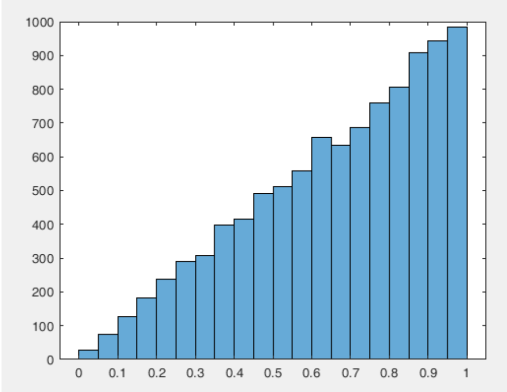
We want a random distibution function

$$f(x) = 2x \quad [0 < x < 1]$$

Then we can find the transformation function

T(r) as follows:

$$r = \int 2x dx = x^2 \quad => \quad x = \sqrt{r} = T(r)$$



- -

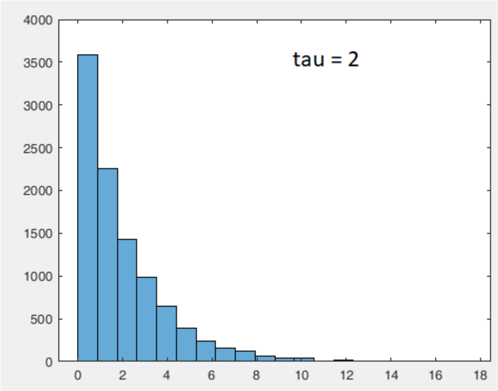
We want a random distibution function

$$f(t) = \frac{1}{\tau} e^{-t/\tau} \quad [0 < t < \infty]$$

Transformation function:

$$r = \int f(t) dt = 1 - e^{-t/\tau} \quad =>$$

$$t = -\tau \ln(1-r) = -\tau \ln(r) = T(r)$$



- In MATLAB, we use **`exprnd(tau)`**
  `exprnd(2)` returns single value with mean 2
  `exprnd(2,5,1)` returns 5x1 matrix (array)
- In Root, we use **`double Exp(double tau)`**
  `rnd.Exp(2)` returns single value with mean 2

Example 7: Uniform point on a sphere

$$\frac{\mathrm{d}p}{\mathrm{d}\Omega} = \frac{\mathrm{d}p}{\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi} = \mathrm{const} \equiv k \qquad \frac{\mathrm{d}p}{\mathrm{d}\theta\,\mathrm{d}\phi} = k\sin\theta \equiv f(\phi)g(\theta)$$

Distributions for θ and φ:

$$f(\phi) \equiv \frac{\mathrm{d}p}{\mathrm{d}\phi} = \mathrm{const} = \frac{1}{2\pi}, \qquad 0 \le \phi \le 2\pi$$

$$g(\theta) \equiv \frac{\mathrm{d}p}{\mathrm{d}\theta} = \frac{1}{2}\sin\theta, \qquad 0 \le \theta \le \pi$$

Calculating the inverse of the cumulative distribution we obtain:

$$\phi = 2\pi r_1$$

$$\theta = \arccos(1 - 2r_2) \qquad [\mathrm{as}\ G(\theta) = \frac{1}{2}(1 - \cos\theta)]$$

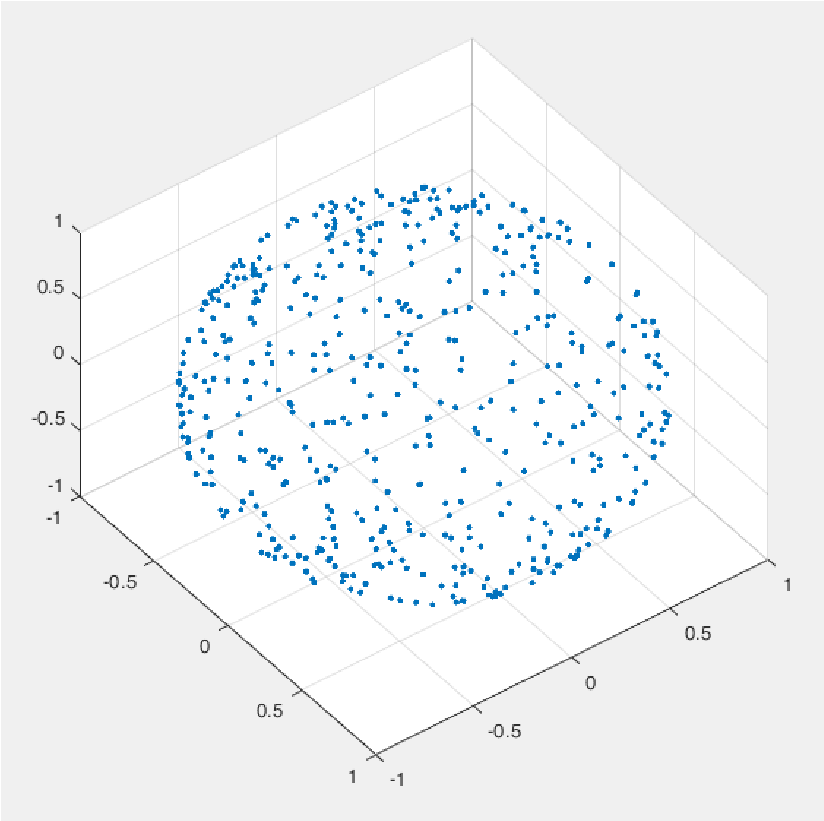
Upshot: φ and cos θ need to be distributed uniformly

```
n = 500;
R = 1;

phi = 2*pi*rand(n,1);
theta = acos(1-2*rand(n,1));

x = R*sin(theta).*cos(phi);
y = R*sin(theta).*sin(phi);
z = R*cos(theta);

plot3(x,y,z,'.')
grid
```



Transformation Method (Discrete RV)

Suppose **X** can take on *n* distinct values **X** = $\{x_1 , x_2 , x_3 , \ldots x_n\}$ with

PDF: $f(x) = \{f_1 , f_2 , f_3 , \ldots f_n\}$ and

CDF: $F(x) = \{f_1 , f_1 + f_2 , \ldots , f_1 + f_2 + f_3 + \ldots + f_n\}$

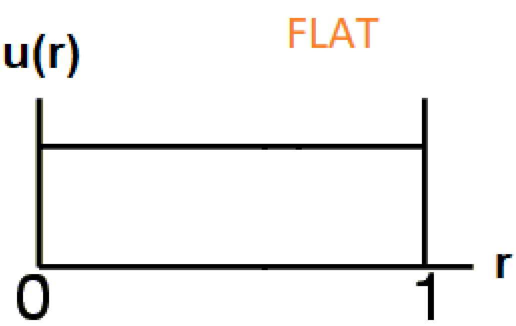
Then to generate a sample value of **X**

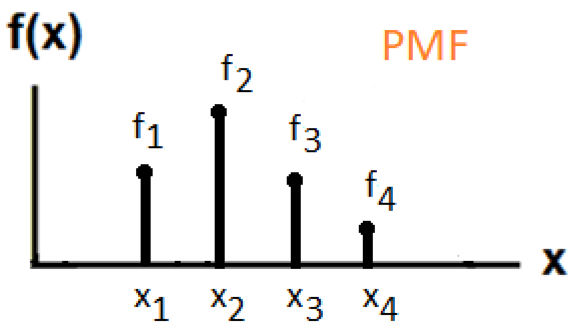
1. Generate uniform ranfom number *r* in the range [0, 1].

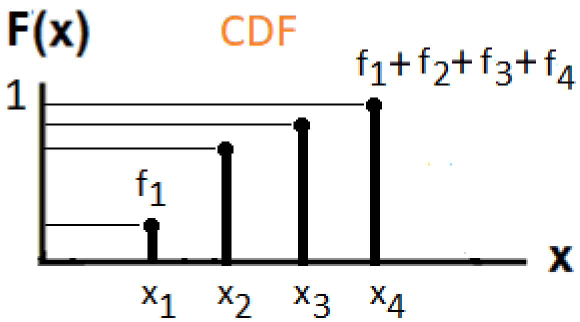
2. for j = 1 to n

Set **X** = x_j if $F_{j-1} < r \leq F_j$

end







Example 8: Branching Ratio

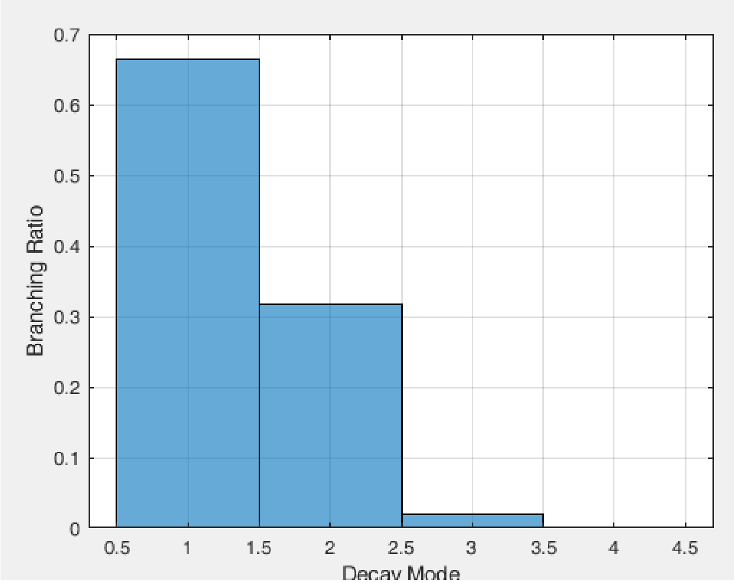
$D^{*\pm}$ decays into 3 different channels.

Select a decay mode randomly.

Mode		*Fraction* (Γ_i / Γ)
Γ_1	$D^0\pi^+$	$(67.7 \pm 0.5)\%$
Γ_2	$D^+\pi^0$	$(30.7 \pm 0.5)\%$
Γ_3	$D^+\gamma$	$(1.6 \pm 0.4)\%$

```
x = [1 2 3];               % decay mode #
f = [0.677 0.307 0.016]; % BR
F = [0.677 0.984 1.000]; % CDF
N = 1000;                  % Look at 1000 decays
X = zeros(N,1);

for i = 1:N
    r = rand;
    for j = 1:3
        if j==1 && r <= F(j)
           X(i) = x(j);
        elseif j > 1 && r > F(j-1) && r <= F(j)
           X(i) = x(j);
        end
    end
end
histogram(X,3,'BinEdges',0.5:4.5,'Normalization','pdf')
xlabel('Channel')
ylabel('Branching Ratio')
grid on
```



Rejection Method

The Transformation Method is useful when the function T(*r*) can easly be evaluated. However, there are cases when desired distribution may not be known in analytic form.

Two examples are as follows:

Gaussian fuction : $f(x) = e^{-x^2}$

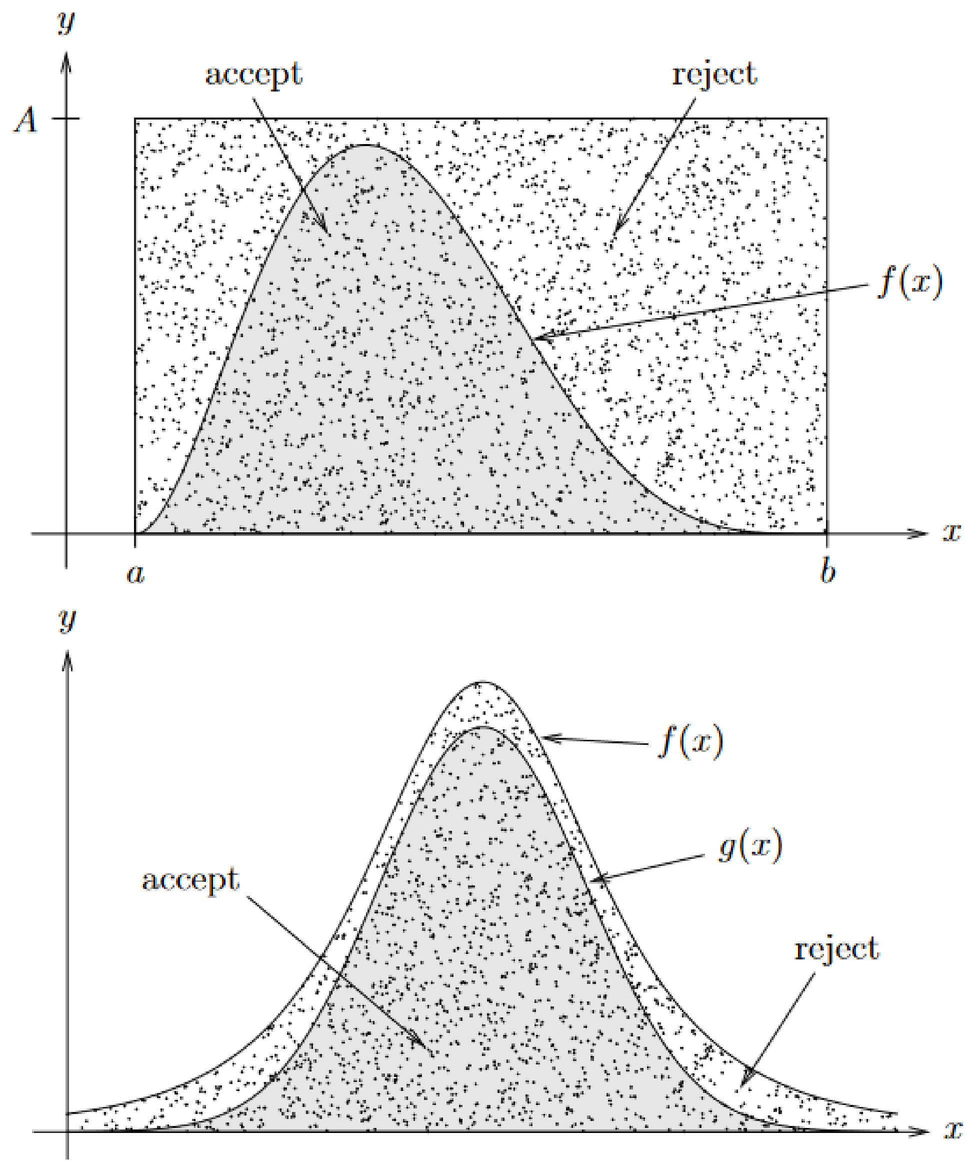
Quadratic function : $f(x) = 1 + x^2$

Such problems can be handled with algorithm known as **Rejection Method.**

It has the advantage of being able to create a distribution for *any function*.

Rejection Method

- Algorithm
  - Generate random number *x* uniformly between *a* and *b*
  - Generate second random *y* number uniformly between 0 and *A*
  - Accept *x* if *y* < f(*x*)
  - Repeat many times
- The efficiency of this algorithm can be quite small
- Improvement possible by choosing a majorant, i.e., a function which encloses *g*(x) and whose integral is known ("importance sampling")



Example 9: Maxwell Distribution in C++

```
double Maxwell_Boltzman(double m, double T){
  //------------------------------------------------------------------
  // Returns, for an atom of mass m (kg) and temperature T (K),
  // a velocity in m/s which is randomly selected from
  // a Maxwell-Boltzman Distribution function using Rejection Method.
  //------------------------------------------------------------------
  double const kB = 1.38e-23;
  double kT    = kB*T;
  double C     = M_PI*sqrt(2.0) * pow(m/(M_PI*kT),1.5);
  double vp    = sqrt(2.0*kT/m);
  double vmin = 0.0;
  double vmax = 10*vp;
  double fmax = C * vp*vp * exp(-1.0), Ptest, fmb;

   // Rejection algorithm
   while(1)
   {
     double r = rnd.Uniform();
     double v = rnd.Uniform();

     Ptest = fmax*r;
     v     = vmin + (vmax-vmin)*v;
     fmb   = C * v*v * exp(-0.5*m*v*v/kT);

     if (fmb > Ptest) return v;
   }
 }
```

$$f(v) = \left[\frac{m}{2\pi kT}\right]^{\frac{3}{2}} 4\pi v^2 \exp\left(-\frac{mv^2}{2kT}\right)$$

Example 9: Maxwell Distribution in Matlab

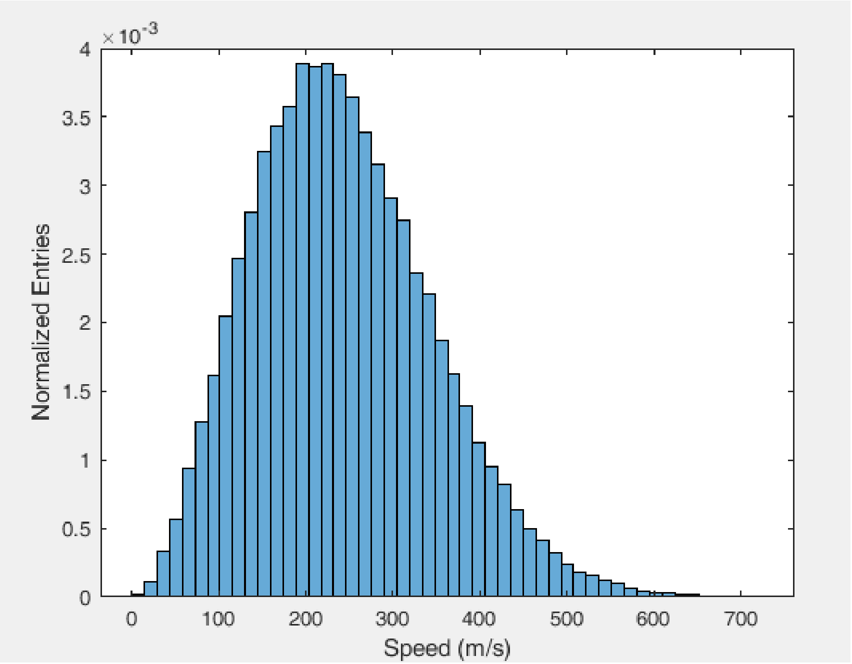
```matlab
clear;
T = 300;      % K,  room temperature
m = 1.8e-25; % kg, mass of silver atom

kB = 1.38e-23; % Boltzman constant
kT   = kB*T;
C    = pi*sqrt(2.0) * (m/(pi*kT))^1.5;
vp   = sqrt(2.0*kT/m); % peak velocity
vmin = 0.0;
vmax = 20*vp;
fmax = C * vp*vp * exp(-1.0);
n    = 100000; % number of random numbers
r    = zeros(n,1);

for i = 1:n
    % Rejection algorithm
    while 1
      Ptest = fmax*rand;
      v     = vmin + (vmax-vmin)*rand;
      fmb   = C * v*v * exp(-0.5*m*v*v/kT);
      if fmb > Ptest
          break;
      end
    end
    r(i) = v; % random value from function
end

histogram(r,50,'Normalization','pdf')
xlabel('Speed (m/s)')
ylabel('Normalized Entries')
```

$$f(v) = \left[\frac{m}{2\pi kT}\right]^{\frac{3}{2}} 4\pi v^2 \exp\left(-\frac{mv^2}{2kT}\right)$$



Monte Carlo Integration

Naïve Monte Carlo integration:

x_i: uniformly distributed random numbers

uniform distribution in $[a, b]$

$$\underbrace{\int_a^b f(x)\,\mathrm{d}x}_{=:\,I} = (b-a)\int_a^b f(x)u(x)\,\mathrm{d}x = (b-a)\langle f(x)\rangle \approx \underbrace{(b-a)\cdot\frac{1}{n}\sum_{i=1}^{n} f(x_i)}_{=:\,\hat{I}}$$

Typical Error (Standard deviation) $\sigma[I] = \dfrac{b-a}{\sqrt{n}}\sigma[f]$

Not efficient in 1D integrals. Very good at higher dimensions.

Exercise: Find volume of the intersection of a cone and a torus

- *Hard to solve analytically*
- *Easy to solve by scattering points homogeneously inside a cuboid containing the intersect.*

Example 10:

Evaluate the following integrals via MC integration method.

(a) $\int_0^{\pi} \sin(x)\, dx = 2.0$

(b) $\int_0^{2\pi} \int_0^{\pi} \sin(\theta) d\theta\, d\phi = 4\pi \approx 12.5664$

```
% solution of (a)
n = 10000;
s = 0;
for i=1:n
    x = pi*rand;
    f = sin(x);
    s = s + f;
end
integ = (pi-0) * s/n
```

```
% solution of (b)
n = 10000;
s = 0;
for i=1:n
    theta = pi*rand;
    phi   = 2*pi*rand;
    f = sin(theta);
    s = s + f;
end
integ = (2*pi-0)*(pi-0) * s/n
```

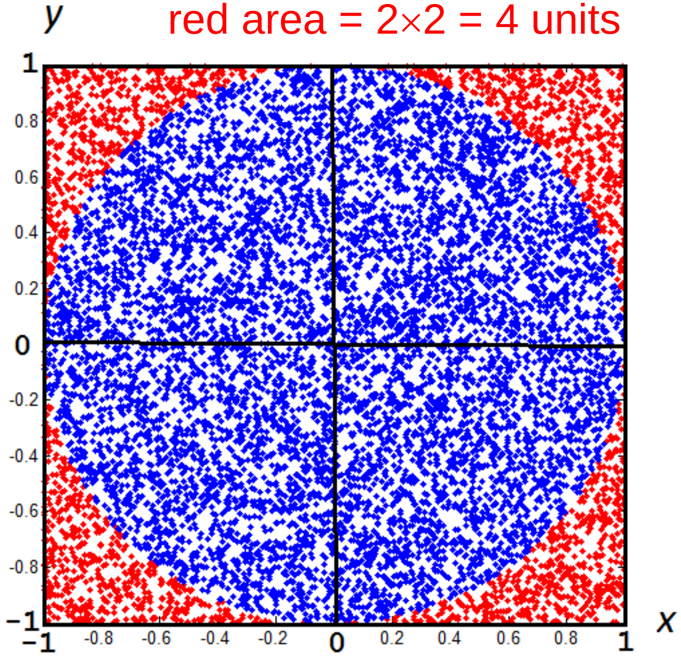
Example 11: Estimating π using MC

Populate a square with *n* randomly-placed points [the blue and red points]

$-1.0 \le x < +1.0$
$-1.0 \le y < +1.0$

Count the number of points *m* that lie inside a circle of unit Radius [the blue points] then $m / n \approx \pi / 4 \Rightarrow \pi \approx 4\, m / n$

blue area = $\pi r^2 = \pi 1^2 = \pi$ units
red area = $2 \times 2 = 4$ units




```
n = 100000;
x = 2*rand(n,1)-1;
y = 2*rand(n,1)-1;
m = sum( x.^2 + y.^2 < 1);
disp(4*m/n)
```

Convergence:

n	
10^3	3.168
10^4	3.1692
10^5	3.14836
10^6	3.140310
10^7	3.1419192
π	3.1415927

Some Applications

1. Modelling Cherenkov Radiation Photons
   http://www1.gantep.edu.tr/~bingul/zemax/src/cherenkov.m

2. Simulation of Two-Body Decay
   http://www1.gantep.edu.tr/~bingul/simulation/twoBody

3. Simulation of Stern-Gerlach Experiment
   http://www1.gantep.edu.tr/~bingul/seminar/spin

4. Simulation of Fussion Chain Reaction
   http://www1.gantep.edu.tr/~bingul/simulation/fission/

5. Tolerance Analysis of a Lens
   See Section 9.3 at:
   http://www1.gantep.edu.tr/~bingul/ep208/latex/ep208_LectureNotes.pdf

6. Predicting Potential Energy Function in Schrödinger Equation via MC
   coming soon …

References

http://www1.gantep.edu.tr/~bingul/seminar/monte-carlo/page1.html

http://www.columbia.edu/~mh2078/MonteCarlo/Generating_RVars_MasterSlides.pdf

https://cas.web.cern.ch/sites/default/files/lectures/thessaloniki-2018/cas-montecarlov6.pdf

https://www.physi.uni-heidelberg.de/~reygers/lectures/2020/smipp/