# *Why Monte Carlo Methods*

First off - by a Monte Carlo method we mean computation with the use of numbers which follow a 'random' sequence according to a probability distribution.

This is necessary for the simulation of processes which are truly random (quantum mechanics). Other processes are so complicated they appear random (e.g., diffusion, coin flipping, …). Some of the first uses of computers were the simulation of neutron diffusion in WWII, and a lot of the theory was developed at that time (Ulam, von Neumann, Metropolis, Fermi, …). Today, Monte Carlo simulations are used in all branches of science.

Monte Carlo methods are also the best known technique for estimating higher dimensional integrals.

# *Some Examples*

Diffusion - example of a stochastic system. Very large number of ‘’degrees of freedom’’ - cannot know the position and velocities of all particles at one time, so work with probability distributions (statistical mechanics).

Ising model - spin correlations, mean field theory. Monte Carlo method uses a stochastic approach to simulate the exchange of energy between spin system and heat bath (external magnetic field).

Monte Carlo event generators in particle physics – basic physics is quantum in nature, so events populate phase space in a random way. The generator produces data sets according to a quantum mechanical model which provides cross sections (probability density functions in phase space).

# *Some Examples*

Simulation of experiments – particle decays follow a probability density law, interactions of particles in detectors probabilistic, number of electrons produced at the photocathode of your PMT probabilistic, … Chain of probabilistic steps for which a simulation is needed to generate expected distributions.

Integration in higher dimensions or in complicated volumes – in 'real life', often faced with integration problems which cannot be solved analytically (e.g., acceptance of a part of your detector in a complicated geometry, high dimensional integrals) and standard numerical approaches are not applicable (too slow), loop integral calculations, …

Optimization problems – trying to find local and/or global maxima and minima in a complicated, possibly higher dimensional space (e.g., extraction of parameters in fitting of data)

# *Brownian Motion*

Discovered in 1827 by the English botanist Brown, who observed that small particles immersed in a liquid exhibit irregular motion. Mathematical description from the laws of physics by Einstein in 1905, who started with the assumption that the motion was caused by repeated collisions of the molecules with the medium. Subject of intense interest since.

$$\frac{\partial p}{\partial t} = D\frac{\partial^2 p}{\partial x^2}$$

Simulate on a grid with unit spacing along x

# *Brownian Motion–cont.*

Heuristic derivation – start from symmetric random walk:

To get to position x at step n+1, we have to be at either x-1 or x+1 at step n:

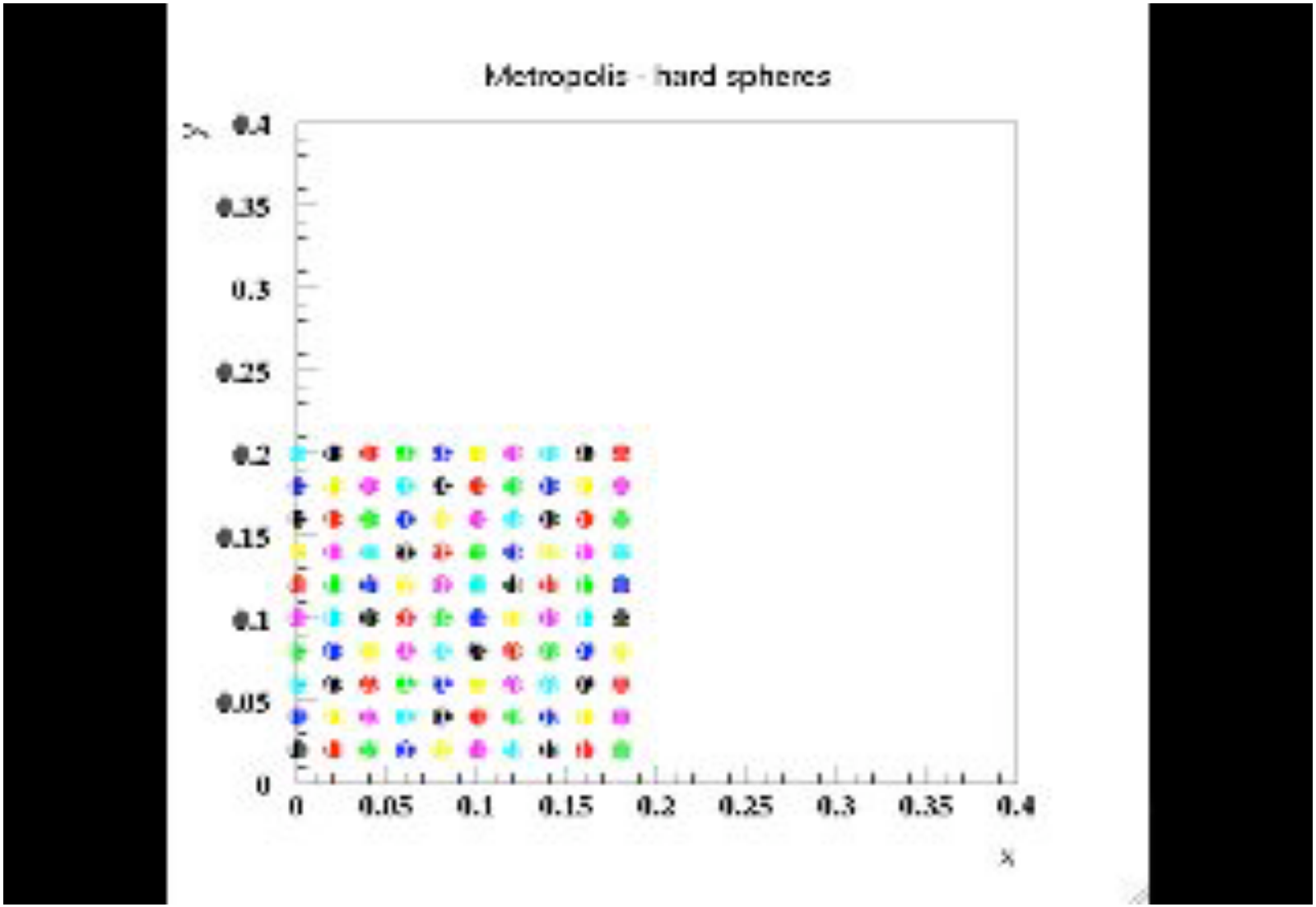
$$P(x,n+1) = \frac{1}{2}P(x-1,n) + \frac{1}{2}P(x+1,n)$$

Now rewrite by subtracting *P(x,n)* from each side

$$P(x,n+1) - P(x,n) = \frac{1}{2}\left[P(x+1,n) - 2P(x,n) + P(x-1,n)\right]$$

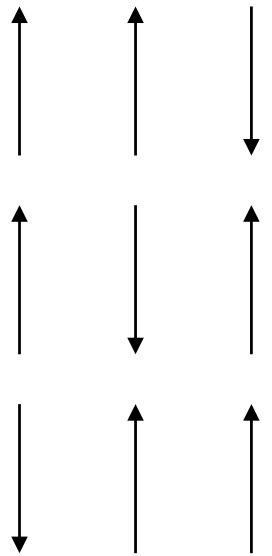
Notice that this looks like a first derivative in time on the LHS of the equation - remember that n is a time variable - and a 2nd derivative of space of the RHS (n fixed). So, we recover the diffusion equation from the random walk. Need physics input to get the units.

# *Diffusion Example*



# *Ising Spin System*

Start with some initial arrangements of spins on a lattice:



Pick one of the spins, and calculate the energy needed to make it flip, $E_{flip}$. Assuming only nearest neighbor interactions, this would be:

$$E_{flip} = -J(\sum_{i,j} s_i s_j - \sum_{i,j} s_i' s_j')$$

If $E_{flip}$ is negative, accept the spin flip. If it is positive, generate a random number distributed flat between [0,1], and compare the the Boltzmann factor. If

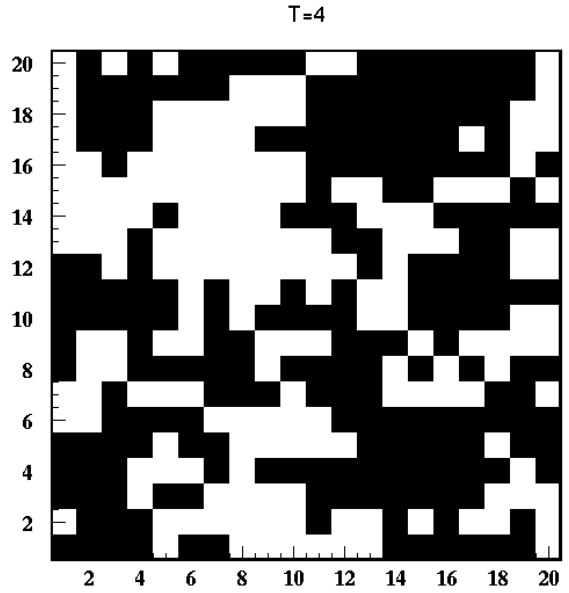
$r < e^{-E_{flip}/k_B T}$ accept the spin flip

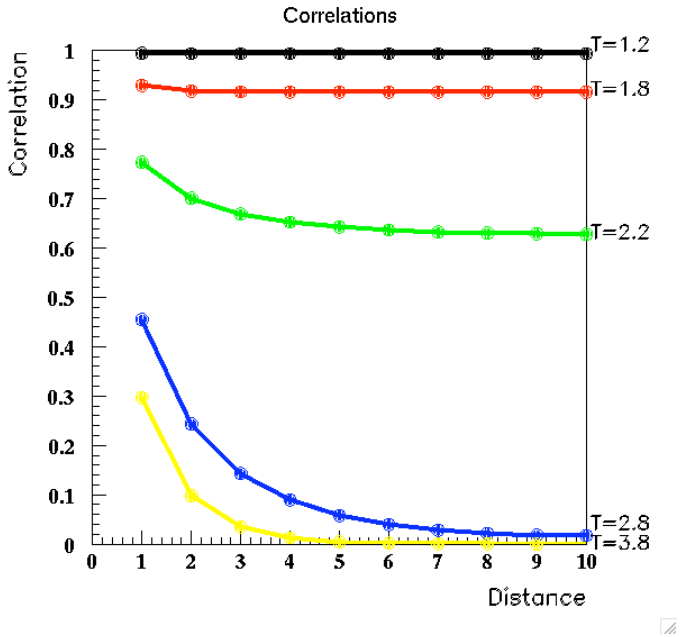
$r \geq e^{-E_{flip}/k_B T}$ reject the spin flip

**Metropolis Algorithm**

# *Ising Spin System*

The equilibrium state is reached when the rates for changing from one spin configuration to another balance. The rate depends on the number of particles in a particular spin state and the probability for a spin flip. The algorithm we have just described yields the correct probabilities to find the system in different energy states.



# *Monte Carlo Integration*

Many numerical techniques for solving integrals:

$$\int_a^b f(x)\,dx = h\sum_{i=1}^{N-1} f(x_i) + \frac{h}{2}\left[f(a)+f(b)\right] - \frac{h^2}{12}\left[f'(b)-f'(a)\right] + O(h^4)$$

Trapezoidal Rule

Precision proportional to spacing, or 1/N where N+1 is number of grid points. Can do better, e.g., using Simpson's rule, extrapolation method, Gaussian quadrature, …, where the error term is $h^4$ or higher. However, for a fixed precision, the number of calculations scales as $N^D$ where D is the dimension of the integral we are estimating.

Look at scaling of MC integration.

# *Monte Carlo Integration*

We want to estimate the following integral:

$$I = \int_D g(\vec{x})\, d\vec{x}$$

where D is a multidimensional volume. Suppose we draw m points in D with an iid (independent, identically distributed) sampling. Then we estimate *I* as

$$\hat{I}_m = \frac{V_D}{m}\left[g(\vec{x}^{(1)}) + g(\vec{x}^{(2)}) + \cdots + g(\vec{x}^{(m)})\right]$$

Law of Large Numbers: $\lim_{m\to\infty} \hat{I}_m = I \quad \text{with probability 1}$

Central Limit Theorem: $\sqrt{m}\left(\hat{I}_m - I\right) \to N(0,\sigma^2) \qquad \sigma^2 = \text{var}\{g(\vec{x})\}$

**The MC method converges as √m independent of the dimensionality**. Note that variance could however be large.

# *Random Numbers*

We now consider how random numbers are generated on the computer. Since these are generated with an algorithm, they are not random, but pseudo-random. This means the distributions of numbers produced by the algorithm should have the properties we expect for uncorrelated random numbers.

Note that having a prescription for generating the random numbers is useful, since we often need reproducible sequences for debugging and reproducibility of programs.

Examples:

- linear congruential generators
- Lagged Fibonacci generator
- …

Follow **Simulation and the Monte Carlo Method**, R. Rubenstein

# *Linear Congruential Generator*

Calculate the residues, modulo an integer, of a linear transformation:

$$X_{i+1} = (aX_i + c)(\mathrm{mod}\, m), \qquad i = 0,...,n$$

$a$ is the multiplier
$c$ is the increment
$m$ is the modulus

} non-negative integers

$X_0$ is the seed, remaining values completely fixed

Random numbers between (0,1) are obtained via:

$$U_i = \frac{X_i}{m}$$

# *Linear Congruential Generator*

Once a previous number is reached, then the sequence will repeat itself. The maximum number of distinct numbers is therefore *m*. The sequence is periodic, and the period is therefore a key value to be determined.

Example:

$$a = c = X_0 = 3 \quad m = 5$$

$$X_{i+1} = \left(3X_i + 3\right) \bmod(5)$$

$$X_0 = 3, X_1 = 2, X_2 = 4, X_3 = 0, X_4 = 3$$

$$\text{Period } p = 4 \text{ (Repeats after 4 steps)}$$

The best we can do is *p=m*. A full period is achieved if

1. *c* is *relative prime* to *m* (*c* and *m* have no common divisors)
2. *a≡1(mod g)* for every prime factor *g* of *m*
3. *a≡1(mod 4)* if *m* is a multiple of 4

*The Art of Computer Programming: Seminumerical Algorithms,* Vol. 2, D. E. Knuth

# *Caveats*

m=$2^{\beta}$ where $\beta$ represents the word length, guarantees a full period, (other conditions mean *c* should be odd and *a=1(mod 4)*

but, need to be careful if need precision (large # of rns):

*For instance, if an LCG is used to choose points in an n -dimensional space, triples of points will lie on, at most, $M^{1/n}$ hyperplanes. This is due to serial correlation between successive values of the sequence …*

*A further problem of LCGs is that the lower-order bits of the generated sequence have a far shorter period than the sequence as a whole if m is set to a power of 2 …*

From Wikipedia

# *Tests of pseudorandom number generators*

It is important to test the random number generator which you will use for your calculations (simulations), or use a generator which has demonstrated properties. There are many tests one can imagine. The most basic is obviously to see that the values are uniformly distributed (you should compare to the theoretical values for the different moments of the distribution, e.g.). In the following, we look at some distributions generated using the RNDM generator in the CERN Library.

Method has: $c = 0$

$$X_0 = 200000000110604716258$$

$$a = 200000003432772446158$$

$$m = 2^{47}$$

On CDC Computer
On your computer ?

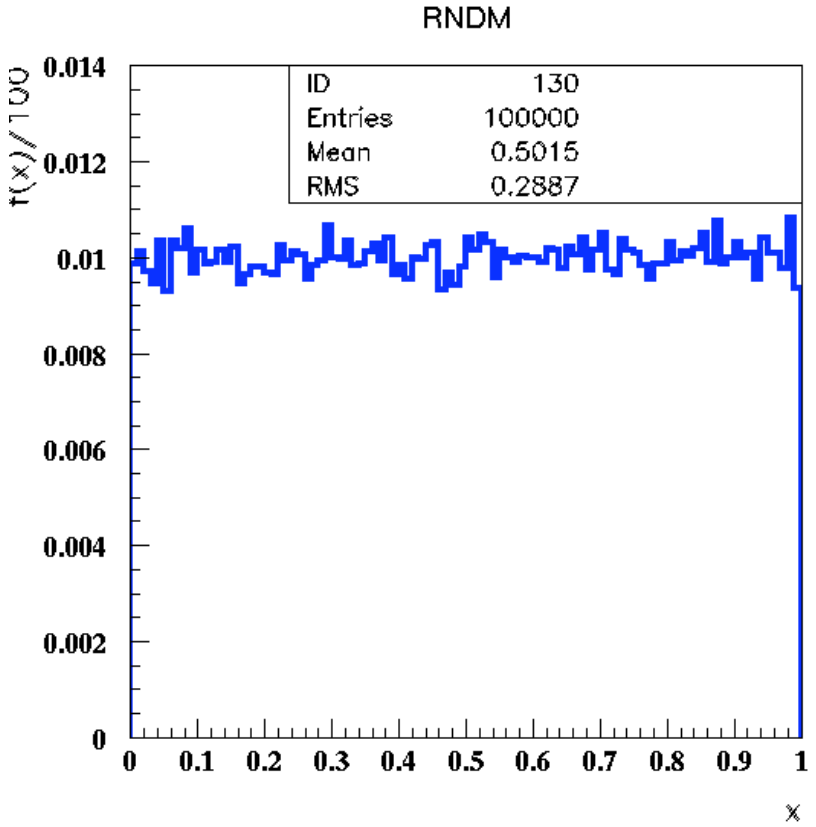
# *Tests of RNDM*

Expectations:

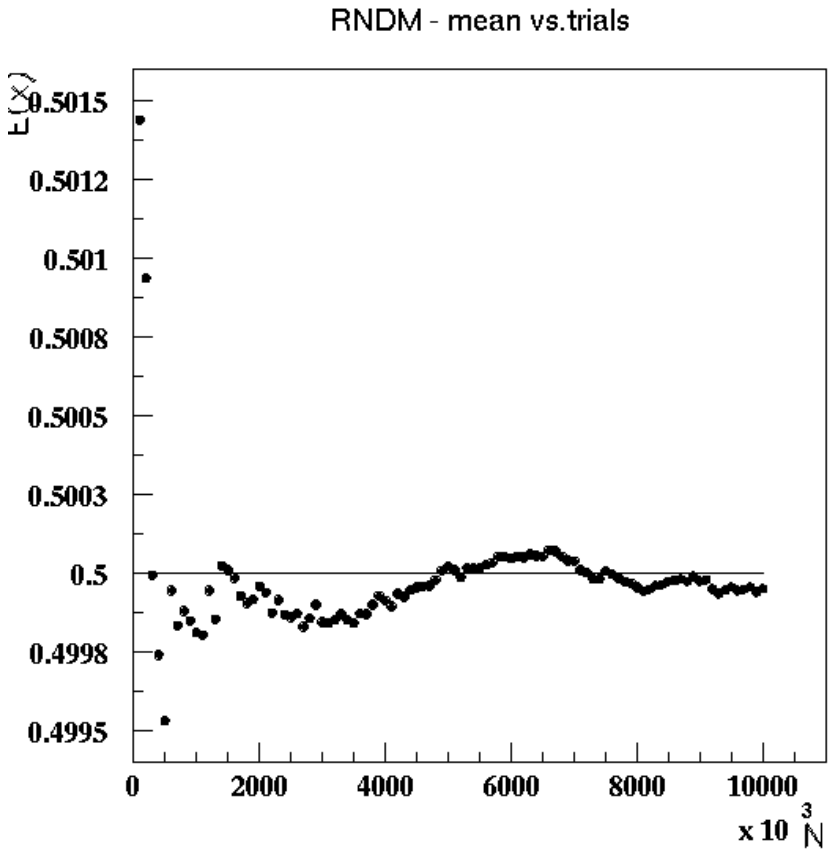
$$E[x] = \int_0^1 x\ f(x)\ dx = \int_0^1 x\ dx = \frac{1}{2}$$

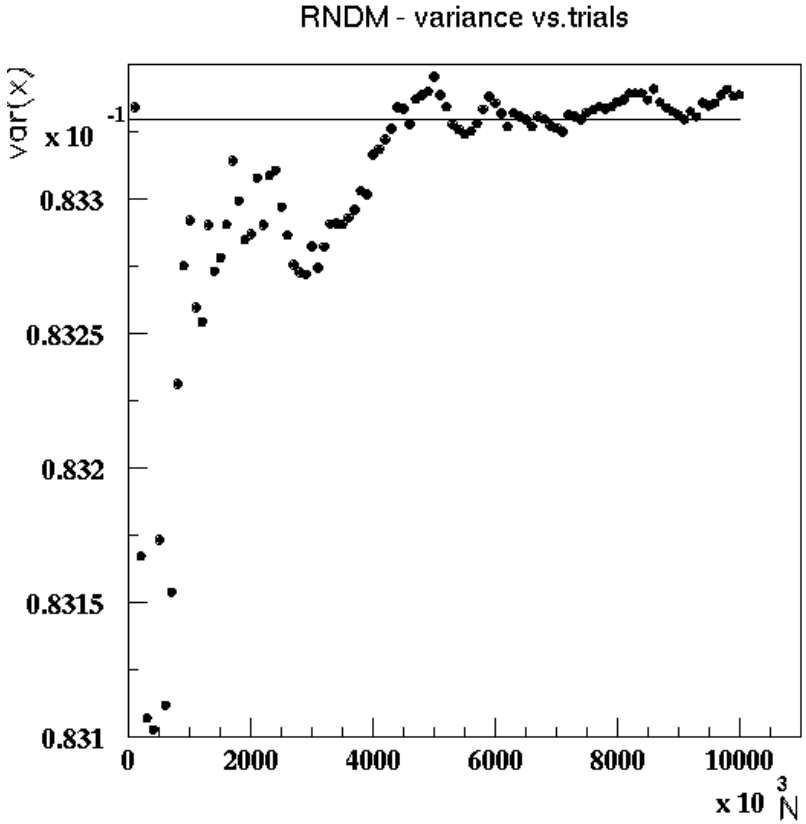
$$m_2 = \int_0^1 \left( x - \frac{1}{2} \right)^2 dx = \frac{x^3}{3} - \frac{x^2}{2} + \frac{x}{4}\Big|_0^1 = \frac{1}{12} = \sigma^2$$

Let us see how our function performs:



# *Mean & Variance*



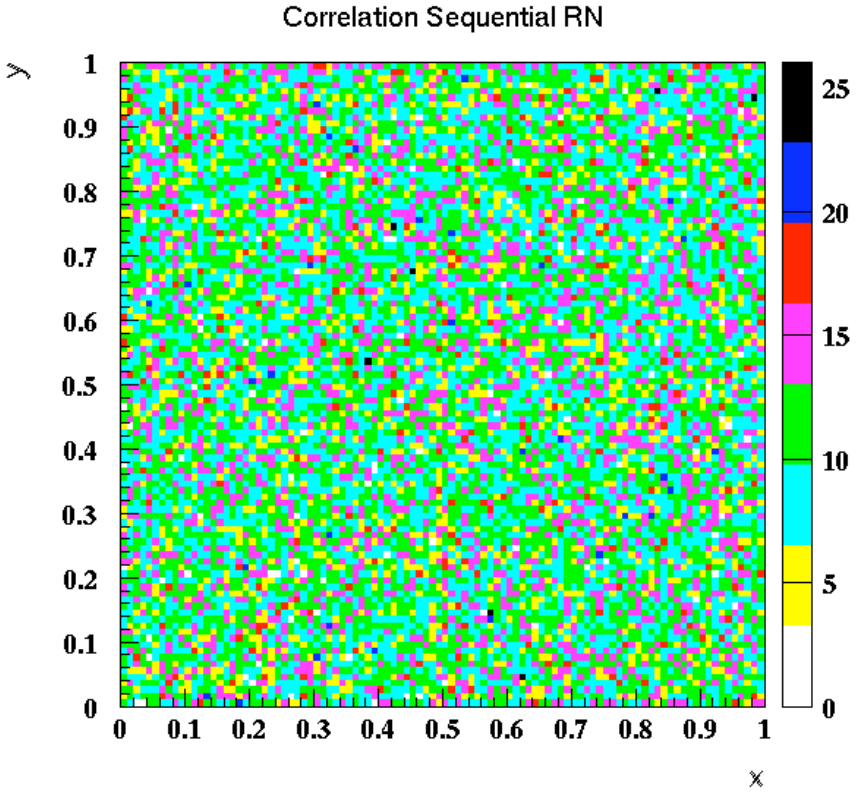



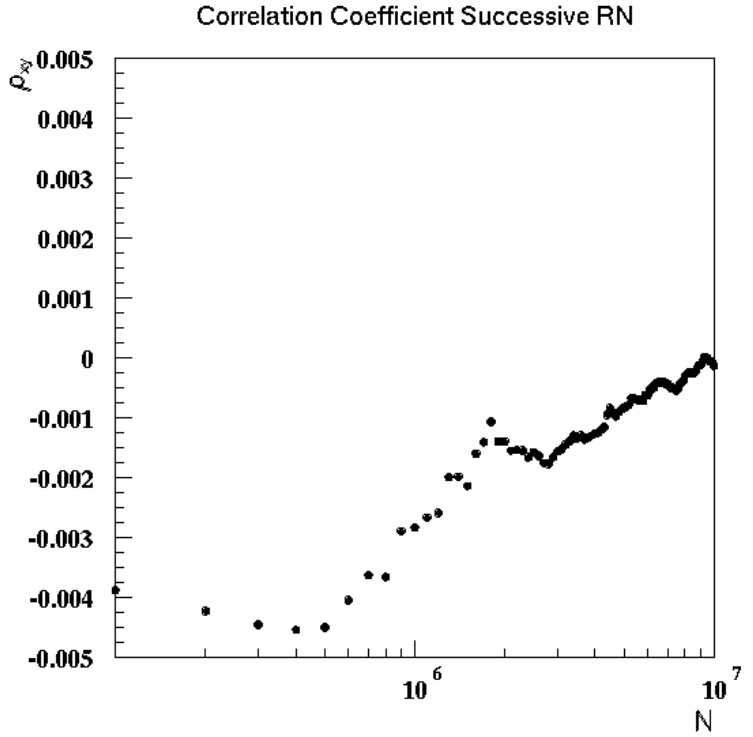
# *Tests of RNDM*

Look at a somewhat more sophisticated quantity, the correlation between successive random numbers. For the correlation coefficient, we expect:

$$\rho_{xy} = \frac{\mathrm{cov}[x,y]}{\sigma_x \sigma_y} = \frac{E[xy] - \mu_x \mu_y}{\sigma_x \sigma_y} = \frac{E[xy] - 1/4}{1/12}$$

$E[xy] = E[x]E[y]$ and $\rho_{xy} = 0$



# *Exercise with Cumulative Distribution Function*

What is the probability density for *xy*, if they are uniformly distributed and independent ?

$$F(a) = \Pr(xy \le a) = \int_0^a f(z)\, dz \qquad \text{where } z = xy$$

$$xy \le a \quad \text{two cases: } x \le a \quad 0 \le y \le 1$$

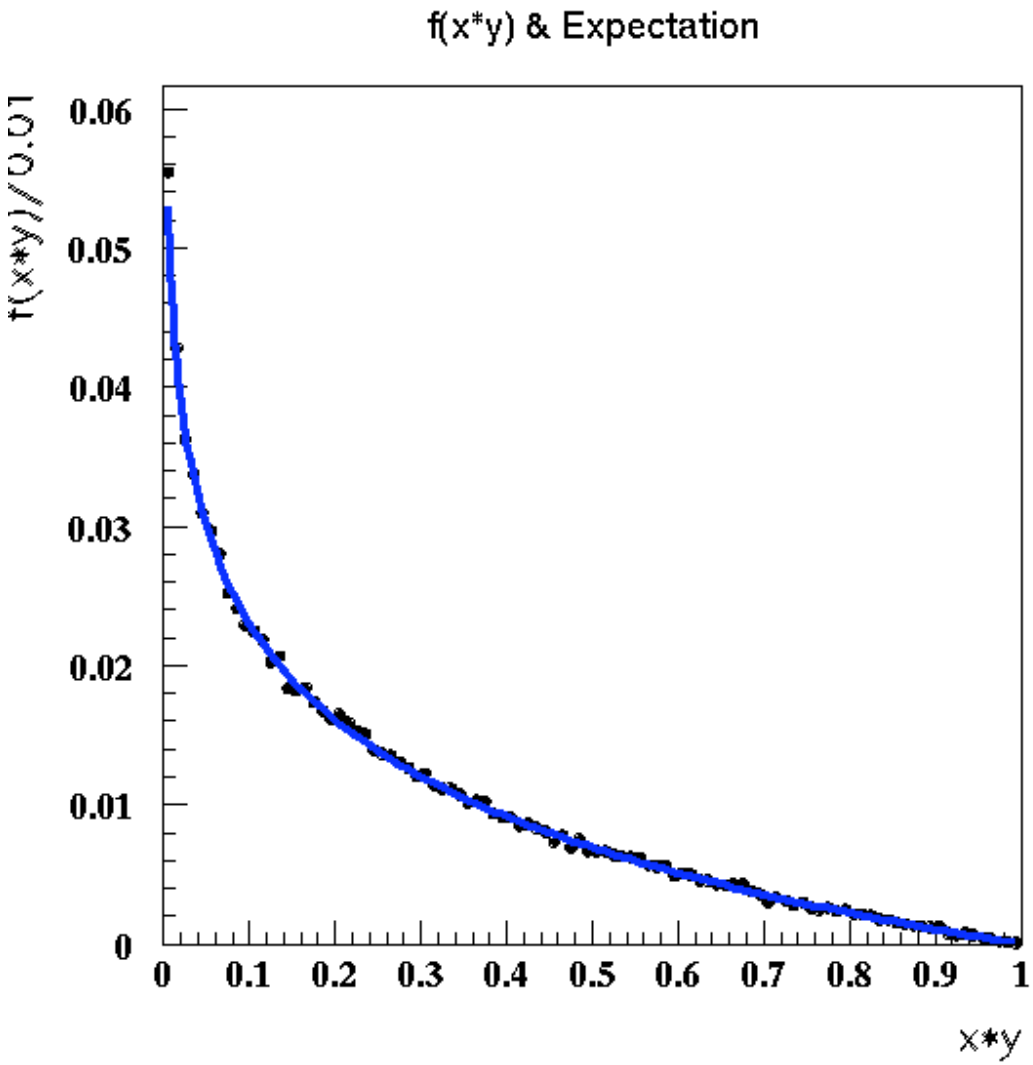
$$x > a \quad y \le a/x$$

So,

$$F(a) = \int_0^a \int_0^1 dy\, dx + \int_a^1 \int_0^{a/x} dy\, dx = a + \int_a^1 a/x\, dx = a + a\ln x\Big|_a^1 = a - a\ln a$$

To get the pdf, we differentiate: $f(z) = \dfrac{dF(z)}{dz} = 1 - \ln z - 1 = -\ln z$

# *Example*


f(x*y) & Expectation
f(x*y)//0.01
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x*y

# *Kolmogorov-Smirnov test*

Define the cumulative distribution function for the sample and compare with the expected:

$$F_N(x) = \frac{\sum_{i=1}^{N} \mathrm{I}_{(-\infty,x)}(X_i)}{\mathrm{N}} \quad \text{where } \mathrm{I}_{(-\infty,x)}(X) = \begin{cases} 1, \text{ if } -\infty < X \le x \\ 0, \text{ otherwise} \end{cases}$$

Look at the max deviation of this from the expected cdf:

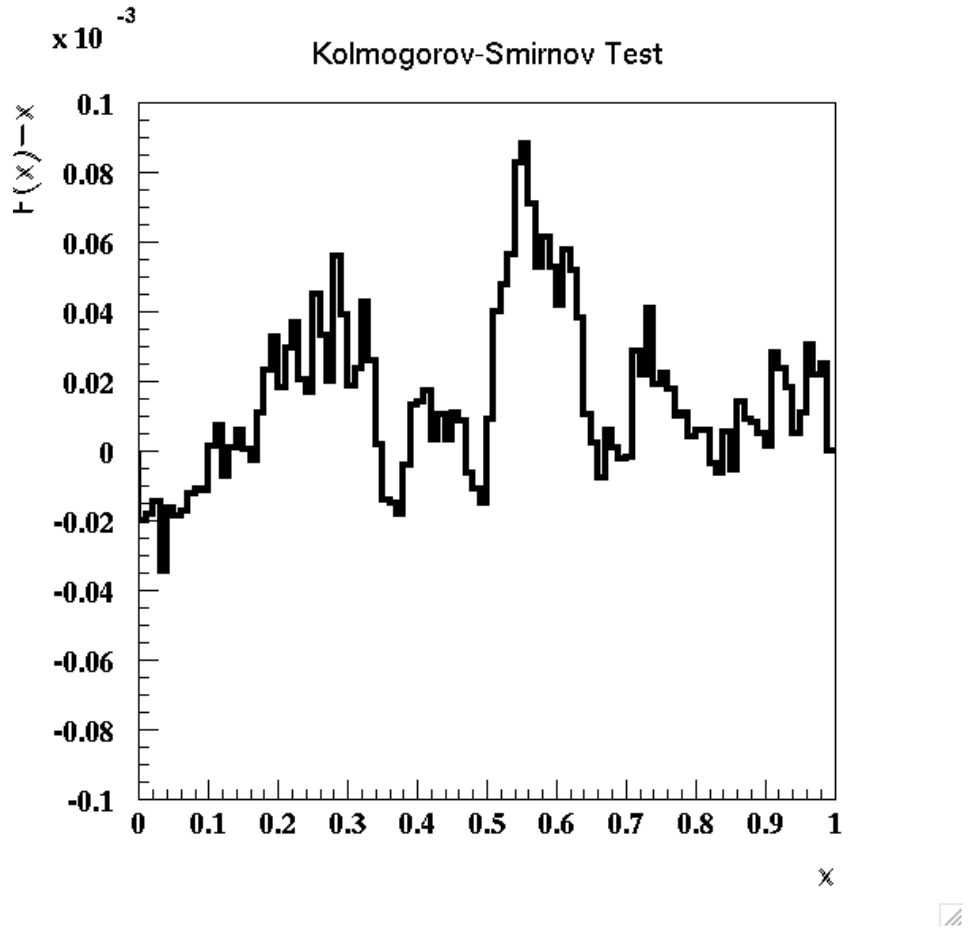
$$D_N = \sup_{-\infty < x < \infty} \left| F_N(x) - F_X(x) \right|$$

$D_N$ should be within a certain value if $F_N$ is really from $F_X$. Expected results are tabulated. Note that for a flat distribution between (0,1), $F_X = x$

# *Kolmogorov-Smirnov Test*

For N>35 or so

| ***Confidence*** | $D_N$ |
|---|---|
| ***20%*** | $1.07/\sqrt{N}$ |
| ***10%*** | $1.22/\sqrt{N}$ |
| ***5%*** | $1.36/\sqrt{N}$ |
| ***2%*** | $1.52/\sqrt{N}$ |
| ***1%*** | $1.63/\sqrt{N}$ |




In our case,
N=5 $10^7$, $1/\sqrt{N}=1.4\ 10^{-4}$
Max deviation is $10^{-4}$, so high confidence that the two distributions agree

# *Exercises*

1. Produce a linear congruential generator which generates uniform random integers between 0,10. Generate a long sequence of numbers and look at mean, variance.

2. Find the cumulative distribution function and the pdf for the product of 3 iid real random numbers with a flat pdf between [0,1) and compare to a simulation.

3. Calculate π by simulating pairs of uniform random numbers and counting the fraction with $r \leq 1$

4. Consider the operation of a Si PM, a square array of cells. Photons are incident on the detector in such a way that each cell has the same probability to get hit. The cells have a 100% efficiency for registering a hit if a photon hits an empty cell, but cannot count more than one photon.
   a) Find an analytic expression for the mean number of cells firing as a function of the number of cells in the SiPM and the number of incident photons.
   b) Write a simulation which produces the distribution of number of cells hit, and compare with the analytic formula.

# *Generating RNs for any Distribution*

So far, we have seen how to generate (pseudo)random numbers in the range [0,1) with a flat probability distribution. We will see how to use this to generate random numbers for any probability distribution. We will consider a couple of different techniques:

- change of variables
- Inverse transform method
- Acceptance-Rejection method

Many special techniques have been developed for individual distributions to speed up the evaluation. We will only consider a couple of examples - check the references for more details.

Also, we look only at generating continuous distributions. Similar techniques apply for discrete distributions → exercises.

# *Change of variables for continuous distribution*

If we have a 1-1 mapping from variables $\vec{x}$ with pdf $f(\vec{x})$, and we want to change to variables $\vec{y} = \vec{y}(\vec{x})$, then we have the resulting pdf

$$g(\vec{y}) = f(\vec{x}(\vec{y}))\,|J|$$

where

$$J_{ij} = \frac{\partial x_i}{\partial y_j}$$

and $|J|$ is the determinant of the Jacobian Matrix. This technique can be used when we have functions we can deal with analytically (rare in 'real' life).

# *Exponential Distribution*

As an example, we consider first generating rns according to an exponential distribution. I.e., we want

$$g(y)dy = e^{-y}dy \qquad 0 \le y \le \infty$$

and we start with $f(x)dx = 1\,dx \qquad 0 \le x < 1$

We want to know *y(x)* which will yield the desired pdf *g(y)*

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

$$\left|e^{-y}dy\right| = |dx| \quad \text{or} \quad x(y) = e^{-y}, \quad \text{and so } y(x) = -\ln x$$

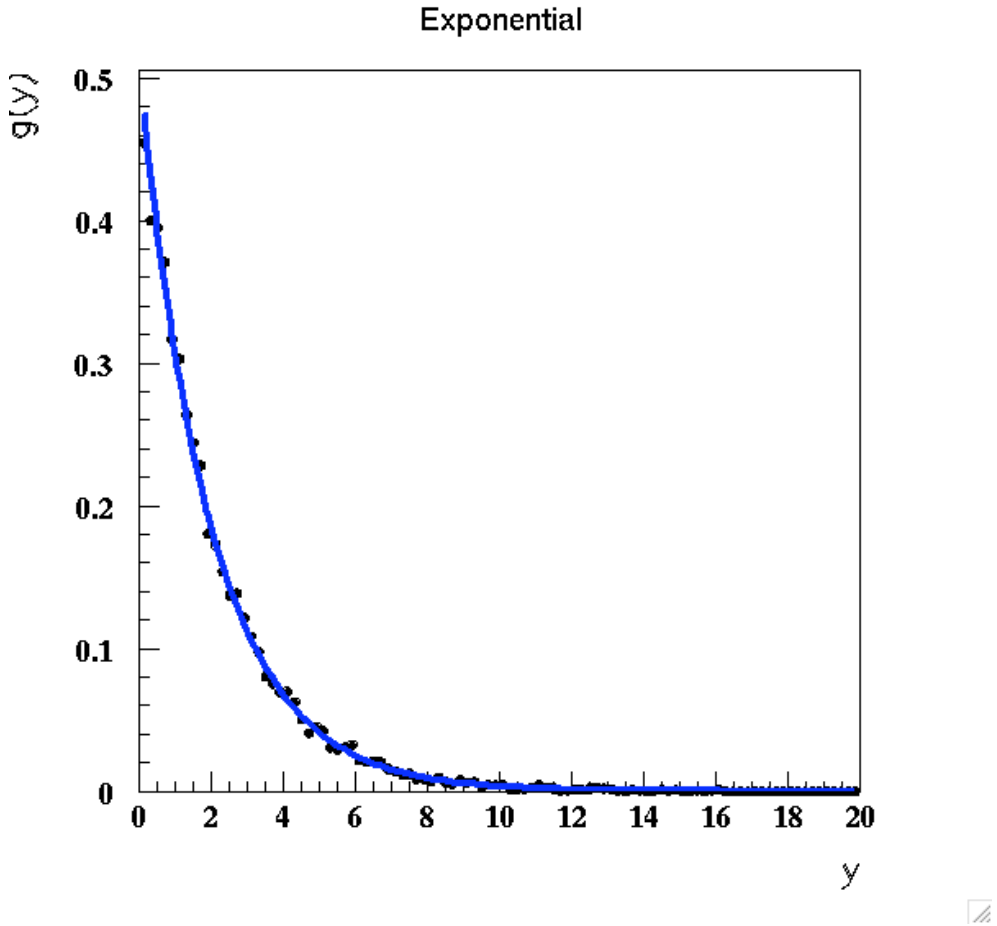
More generally, $g(y) = \frac{1}{\lambda}e^{-y/\lambda} \qquad y > 0, \lambda > 0$

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

# *Exponential Distribution*

Some Fortran code:

```
*
      slope=2.
*
      Call Ranlux(RVEC,Len)
*
      Do I=1,Len
        x=rvec(I)
        y=-slope*alog(x)
        write(10,*) x,y
      Enddo
*
```



# *Gaussian Distribution*

Box-Muller method:

We want $$g(y)dy = \frac{1}{\sqrt{2\pi}} e^{-y^2/2} dy$$

Consider the following construction

$$y_1 = \sqrt{-2\ln x_1}\cos 2\pi x_2 \qquad y_2 = \sqrt{-2\ln x_1}\sin 2\pi x_2$$

where $f(x)dx = dx \qquad 0 < x < 1$

and $x_1, x_2$ are independent

Solving for $x_1, x_2$

$$x_1 = \exp\left[-\frac{1}{2}\left(y_1^2 + y_2^2\right)\right] \qquad x_2 = \frac{1}{2\pi}\arctan\frac{y_2}{y_1}$$

# *Gaussian Distribution*

The Jacobian Determinant is

$$\begin{vmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{vmatrix} = \begin{vmatrix} -y_1 e^{-\frac{1}{2}(y_1^2+y_2^2)} & -y_2 e^{-\frac{1}{2}(y_1^2+y_2^2)} \\ -\frac{1}{2\pi}\left(\frac{y_2}{y_1^2+y_2^2}\right) & \frac{1}{2\pi}\left(\frac{y_1}{y_1^2+y_2^2}\right) \end{vmatrix} = -\frac{1}{2\pi} e^{-\frac{1}{2}(y_1^2+y_2^2)}$$

$$= -\left[\frac{1}{\sqrt{2\pi}} e^{-y_1^2/2}\right]\left[\frac{1}{\sqrt{2\pi}} e^{-y_2^2/2}\right]$$

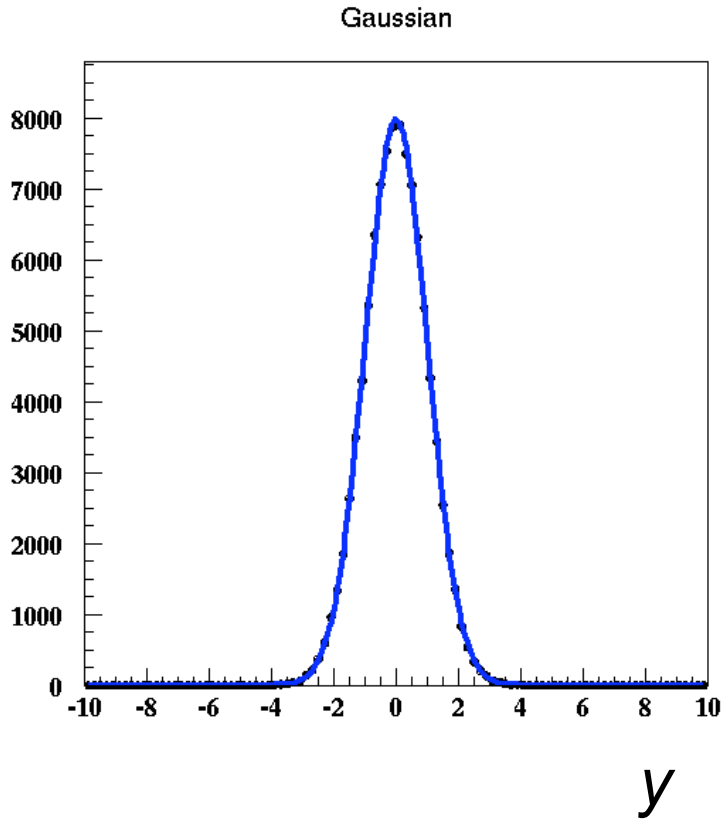
i.e., each of $y_1, y_2$ will independently be distributed according to a Gaussian distribution if we choose

$$y_1 = \sqrt{-2\ln x_1}\cos 2\pi x_2 \qquad y_2 = \sqrt{-2\ln x_1}\sin 2\pi x_2$$

# *Gaussian Distribution*

Some code:

```
*
      Call Ranlux(rvec,Len)
*
      Do I=1,Len,2
        x1=rvec(I)
        x2=rvec(I+1)
        y1=sqrt(-2.*alog(x1))*cos(twopi*x2)
        y2=sqrt(-2.*alog(x1))*sin(twopi*x2)
        write(11,*) x1,x2,y1,y2
      Enddo
*
```



# *RN using Cumulative Distribution Function*

Suppose you don't have an analytic form for the pdf you want to generate (e.g., you may want to generate numbers according to an empirically obtained pdf). How to proceed ?

Using the cdf: $$F(x) = \int_0^x f(x')\, dx'$$

Now suppose *x* is a rn distributed according to *f(x).* *F* is now also a rn. Its distribution is:

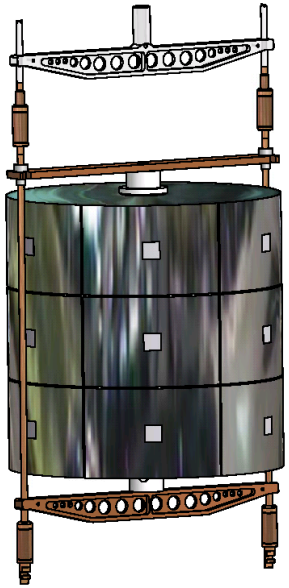
$$g(F) = f(x)\frac{dx}{dF} = 1$$

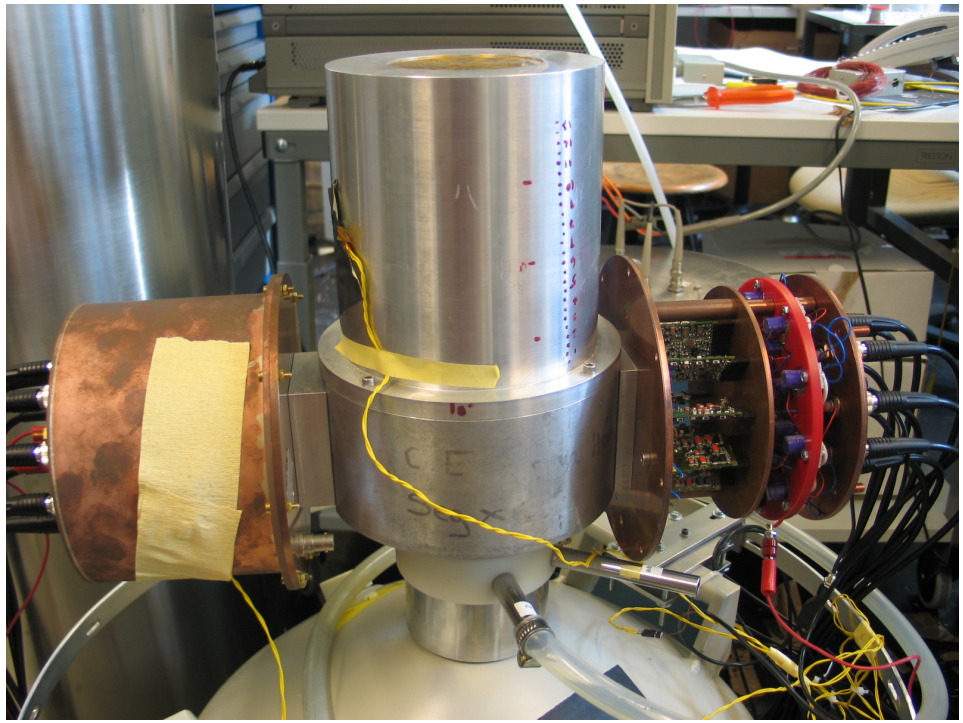
*F* has a uniform distribution between (0,1). So, to get x according to *f(x)*, generate a rn number from the uniform distribution and set it equal to F, and then find the value of *x* such that

$$F(x) = \int_0^x f(x')\, dx'$$

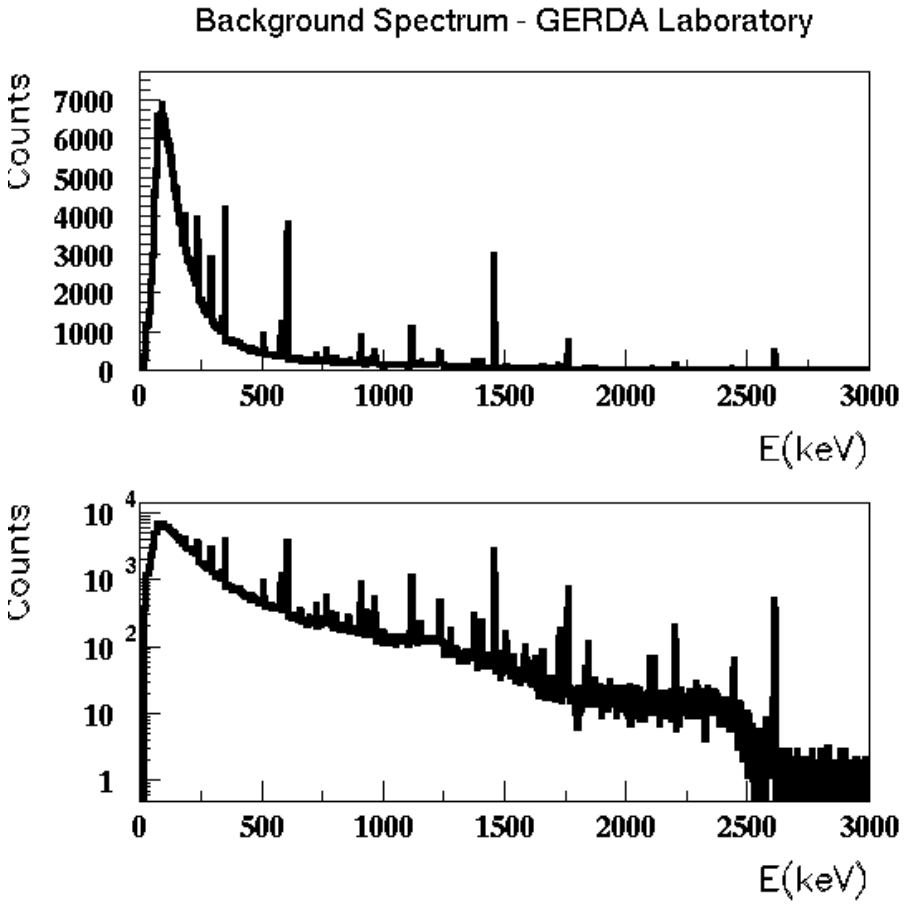
# *RN with cdf*

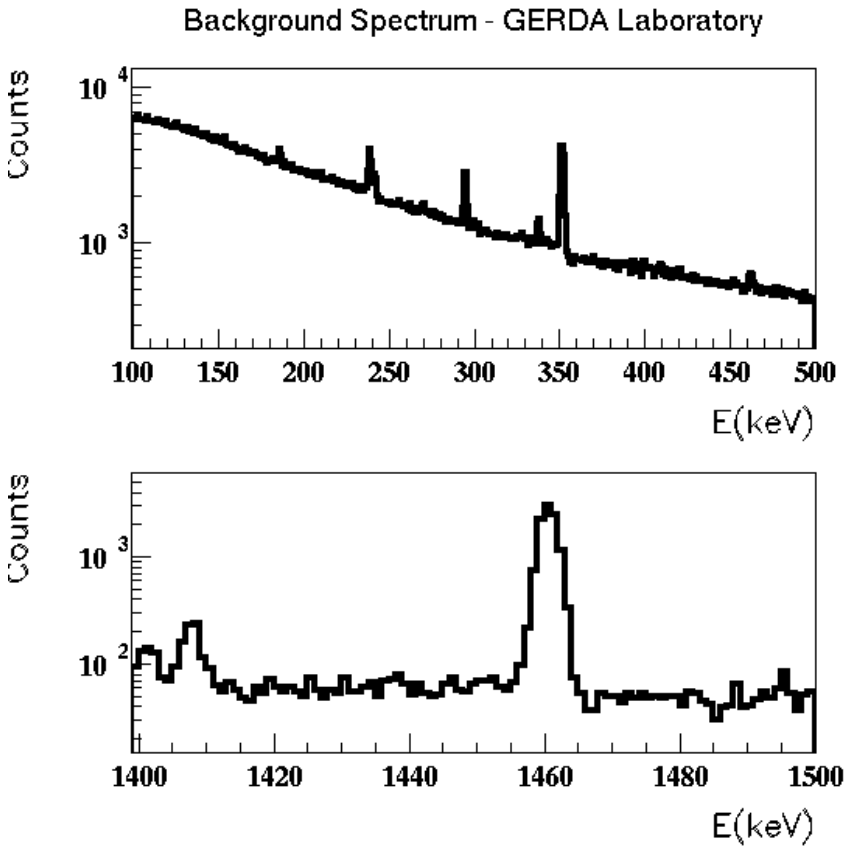
Example: In the GERDA laboratory at the MPI, we are studying the properties of Ge detectors. We put a source close to the detector, and look at the spectrum, which we then compare to our MC simulations. In addition to the source, we also have background counts, which we don't know how to simulate in detail. However, we can measure it, and simulate based on the measured distribution.





# *Measured Background Spectrum*



# *Simulation of Background*

```
      Nsum=0
      Do I=1,3000
        Read(20,*) Energy(I),Counts(I)
        Nsum=Nsum+Counts(I)
      Enddo
* Get the normalized running sum
      CDF(1)=Float(Counts(1))/Nsum
      Do I=2,3000
        CDF(I)=CDF(I-1)+float(Counts(I))/Nsum
      Enddo
* Now generate the background distribution using the cdf and interpolation
      Call Ranlux(RVEC,1000000)
* For each random number, search energy interval from cdf
      Do I=1,1000000
        Write(21,*) rvec(I),-LOCATR(CDF,3000,Rvec(I))
      Enddo
```

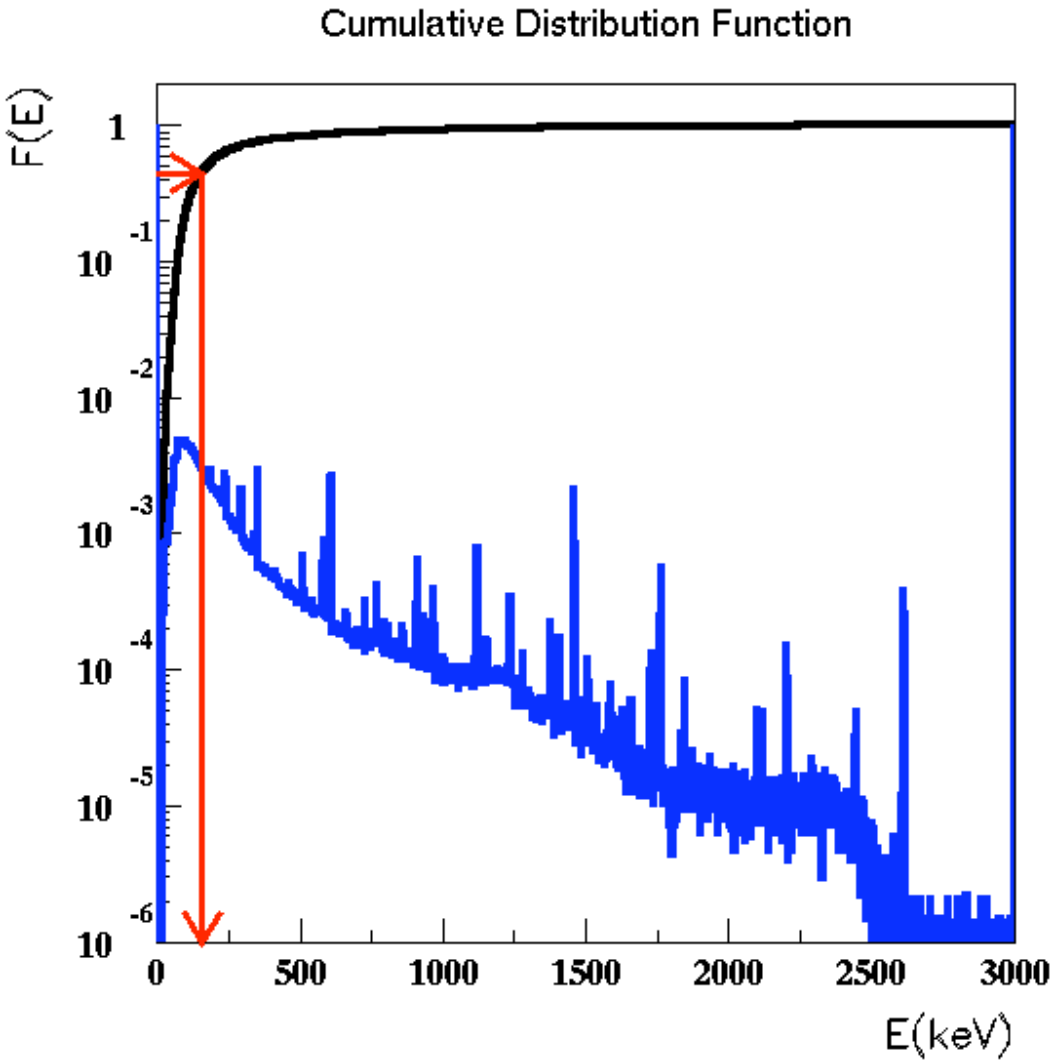
The background is given in the form of a histogram with 3000 bins, from 0-3000 keV

} work out the cdf

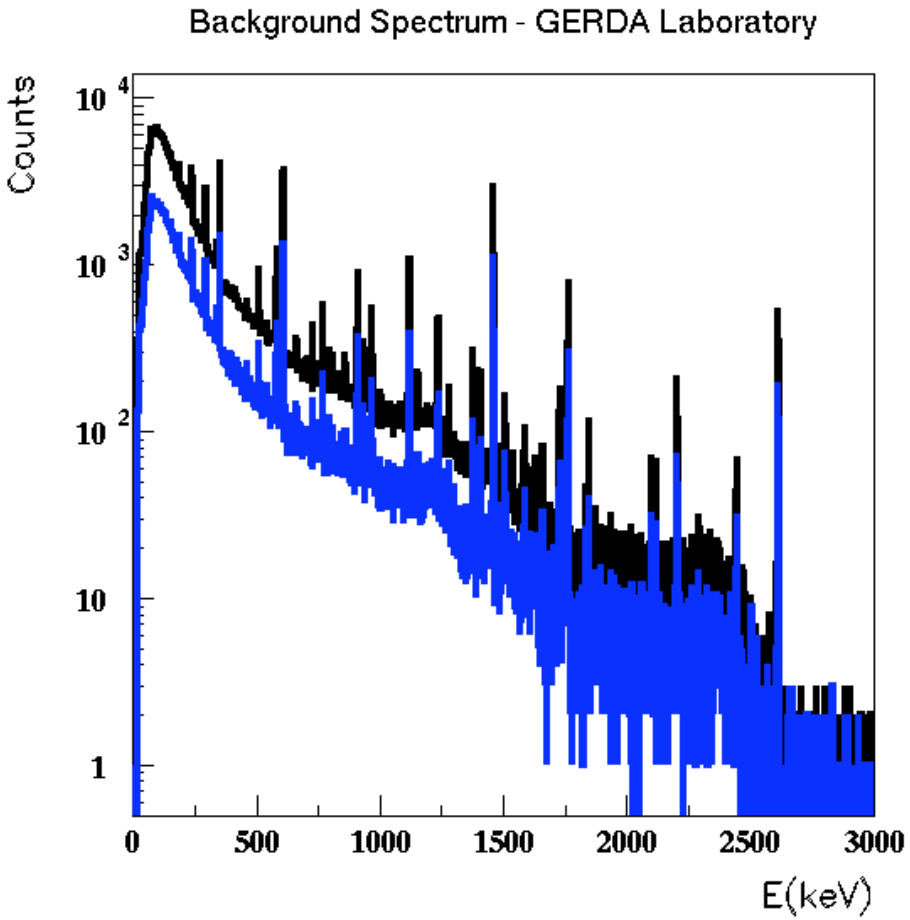
Binary search routine to find bin k (energy interval) for which

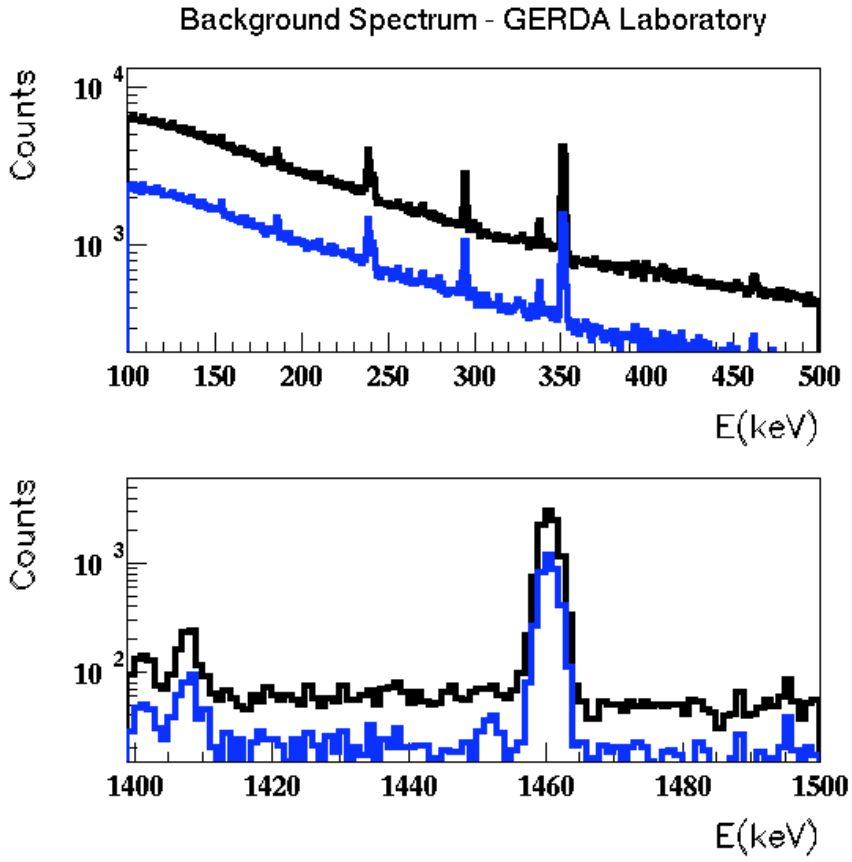
$$cdf(k) \le rvec(I) \le cdf(k+1)$$

# *Background cdf*



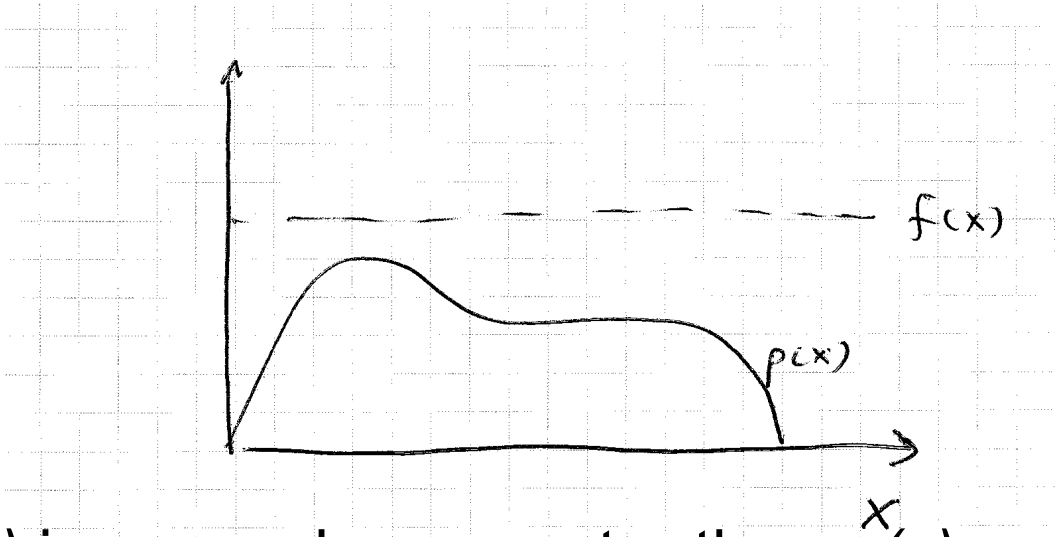
# *Simulated Background*



# *Acceptance-Rejection Method*

We want to generate values of *x* according to *p(x)dx,* but cannot (or prefer not to) use the cumulative distribution function.




Suppose *f(x)* is everywhere greater than *p(x)*, and that we can sample uniformly in the two dimensional area given by *(x,f(x)).* Then, we reject values for which *f(X)>p(X), where X is the randomly chosen value of x.* This will give a distribution in *x* which follows *p(x).*

# *Rejection Method*

Technique:

rewrite $p(x) = Cf(x)g(x)$ where $0 < g(x) \leq 1$

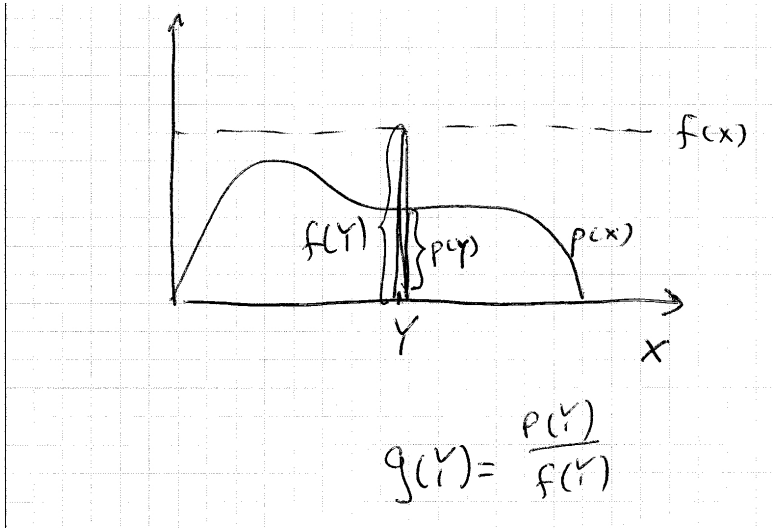
generate $U$ from a uniform distribution between 0,1

and $Y$ according to $f(x)$.

$$g(x) = \frac{p(x)}{f(x)}$$

Then,

1. If $U \leq g(Y)$, then accept and set $X = Y$
2. If $U > g(Y)$, try again



# *Rejection Method*

Note that $\int_{-\infty}^{\infty} p(x)\,dx = 1$

The efficiency of the method will depend on $\int_{-\infty}^{\infty} f(x)\,dx = A$

A fraction *1/A* of trials will be accepted.

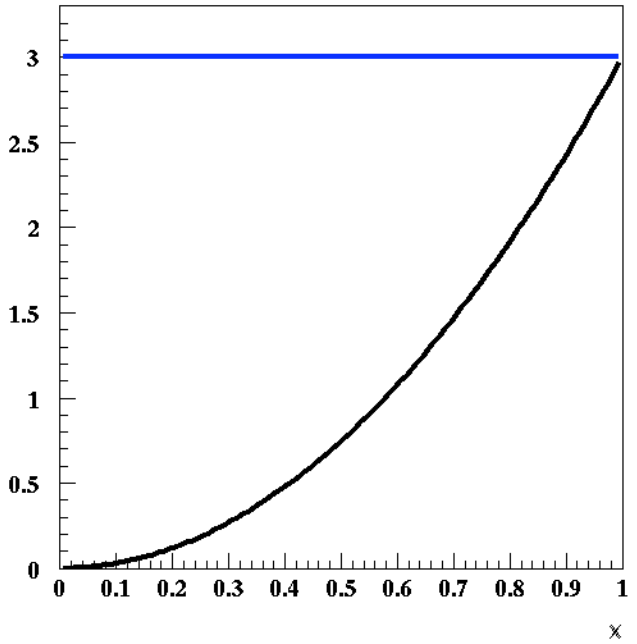
To generate values according to *f(x),* if possible we pick a form where we can calculate the cdf analytically and take the inverse as discussed earlier.

Very common technique. However, in several dimensions, the method can be very inefficient if a good covering function cannot be found.

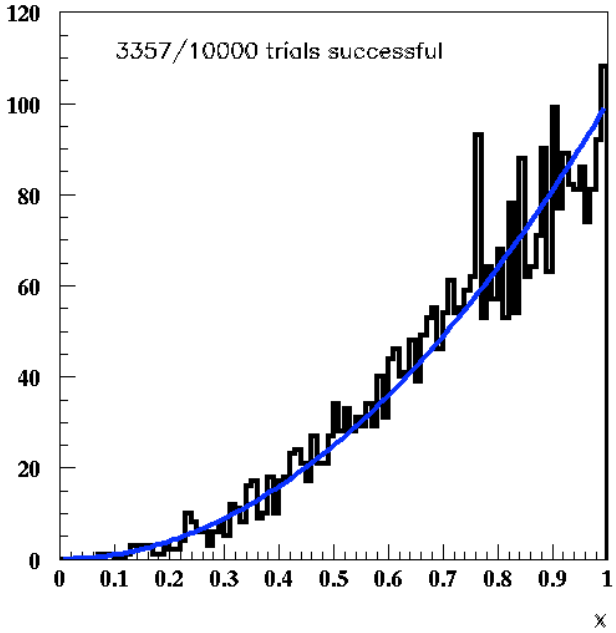
# *Example*

$$p(x) = 3x^2 \qquad 0 \le x \le 1$$

$$f(x) = 3$$




Efficiency will be *1/3*



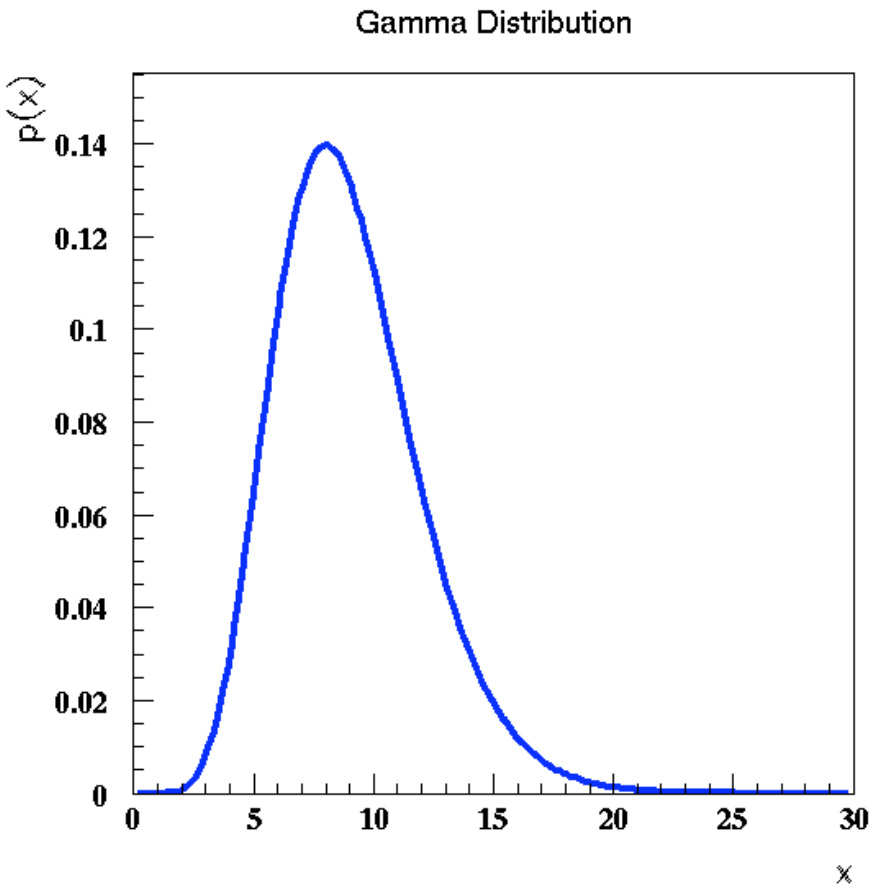
# *Gamma Distribution*

Gamma Distribution: $$p_a(x)dx = \frac{x^{a-1}e^{-x}}{\Gamma(a)}dx \qquad x > 0$$

For *a* an integer, the gamma distribution is the waiting time for the $a^{th}$ event in a Poisson process of unit mean.

For *a=1*, this is just the usual exponential distribution.

Look at *a=9.*



# *Gamma Distribution*

As a covering function, try Lorentzian Distribution

$$f(x)dx = \frac{1}{\pi}\left(\frac{1}{1+x^2}\right)dx$$

The cdf is $$F(x) = \int_0^x \frac{1}{\pi}\left(\frac{1}{1+x'^2}\right)dx' = \frac{\arctan(x)}{\pi}$$

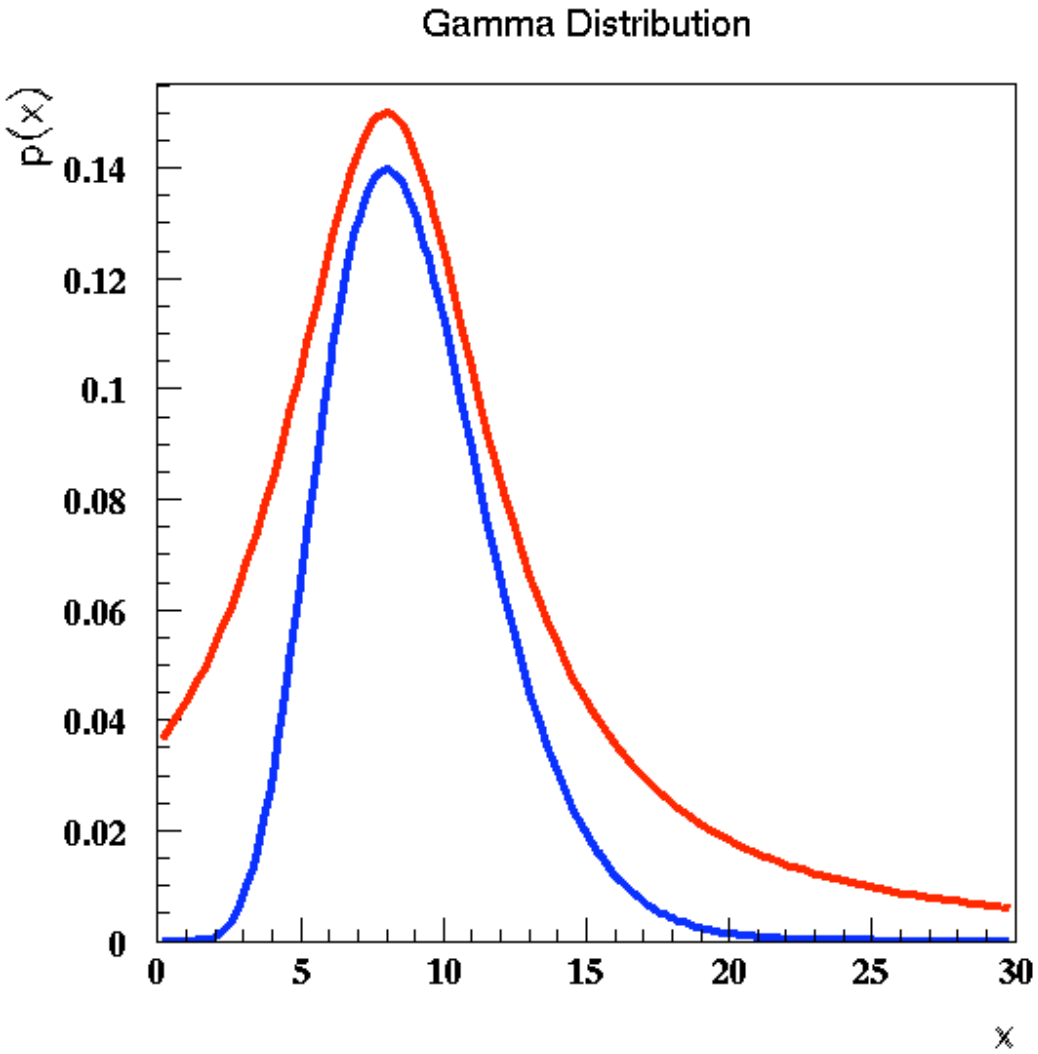
and we generate *x* from the inverse

$$x = F^{-1}(U) = \tan(\pi U)$$

We can scale and shift the distribution as needed:

$$x = a_0 \tan(\pi U) + x_0$$

# *Gamma Distribution*



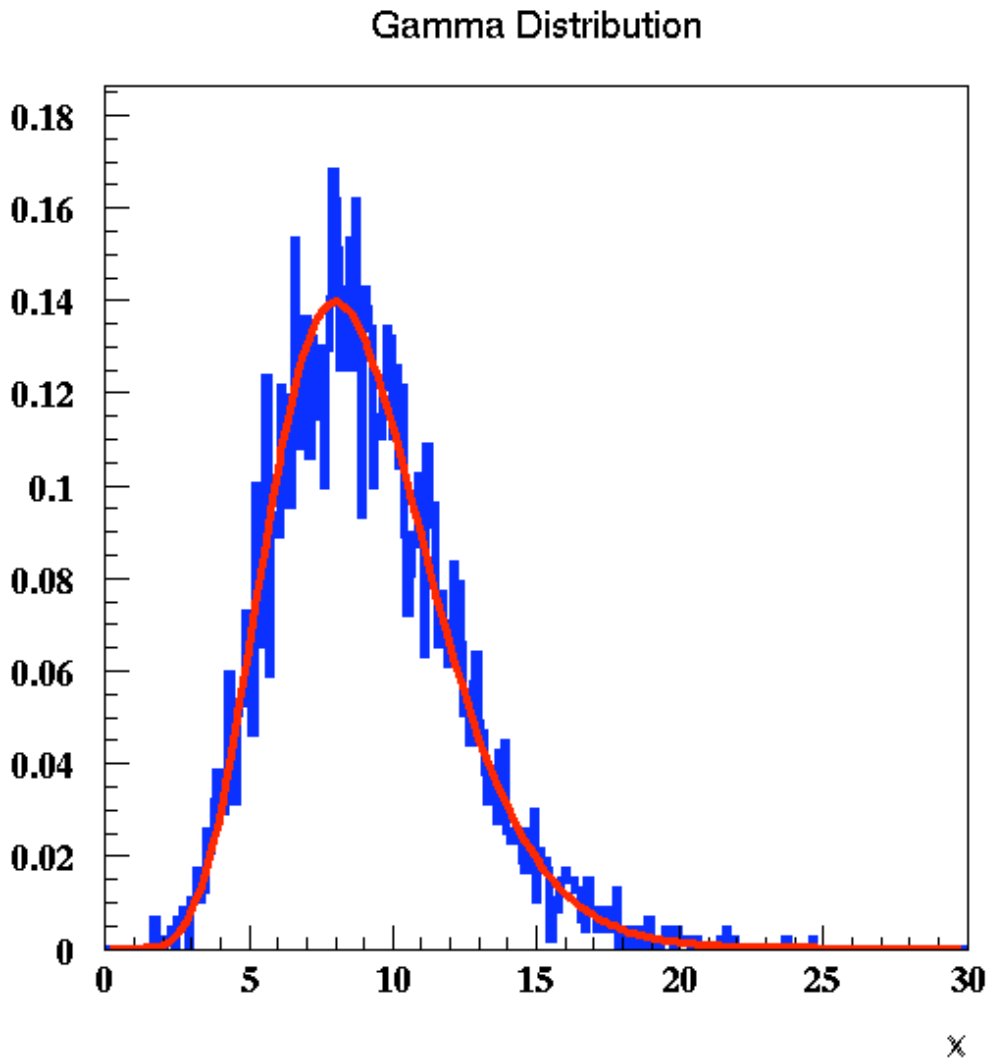

$$f(x) = \frac{0.15}{(1 + 0.05(x - 8.)^2)}$$

# *Gamma Distribution*

```
*
      Call Ranlux(rvec,10000)
      Call Ranlux(rvec1,10000)
*
      Do I=1,10000
        x=sqrt(20.)*tan(3.1415926*rvec(I))+8.
        If (rvec1(I).le.testfun2(x)/cover2(x).and.
     &       x.gt.0..and.x.lt.30.) then
          write (26,*) I,x
        Endif
*
      Enddo
*
```

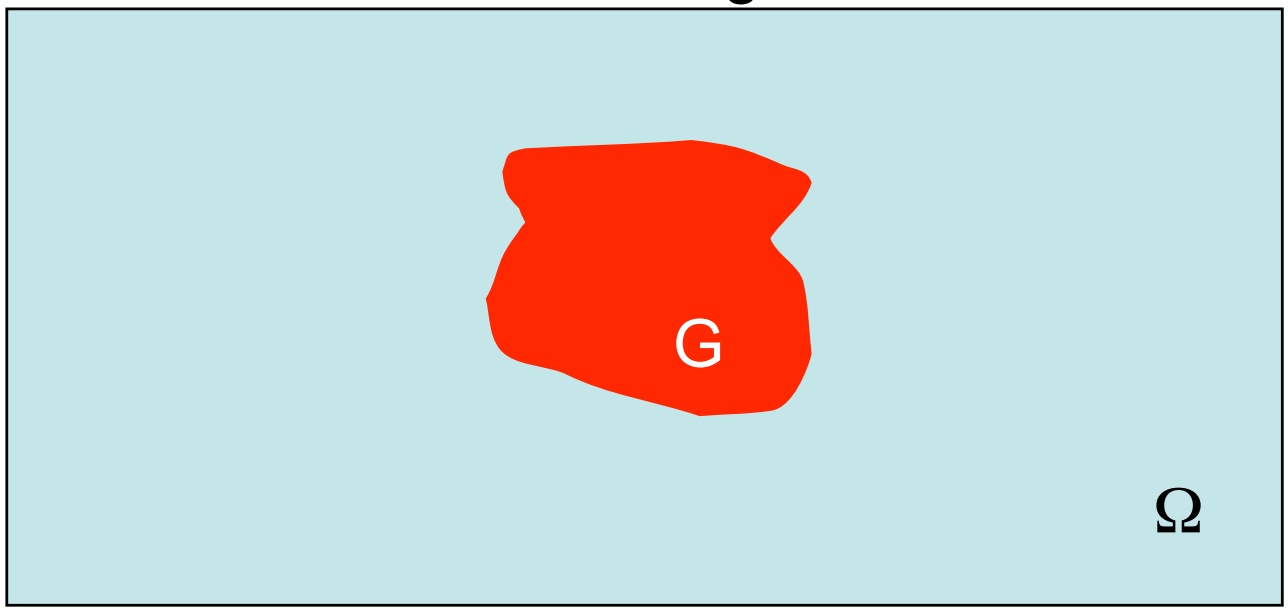
```
      Real Function testfun2(x)
*
      Implicit None
      real x,gamma
*
      testfun2=x**8*exp(-x)/gamma(9.)
*
      return
      end
*
      Real Function cover2(x)
*
      Implicit None
      real x
*
      cover2=0.15/(1+0.05*(x-8.)**2)
*
      return
      end
```

# *Gamma Distribution*



# *Rejection Method in Several Dimensions*

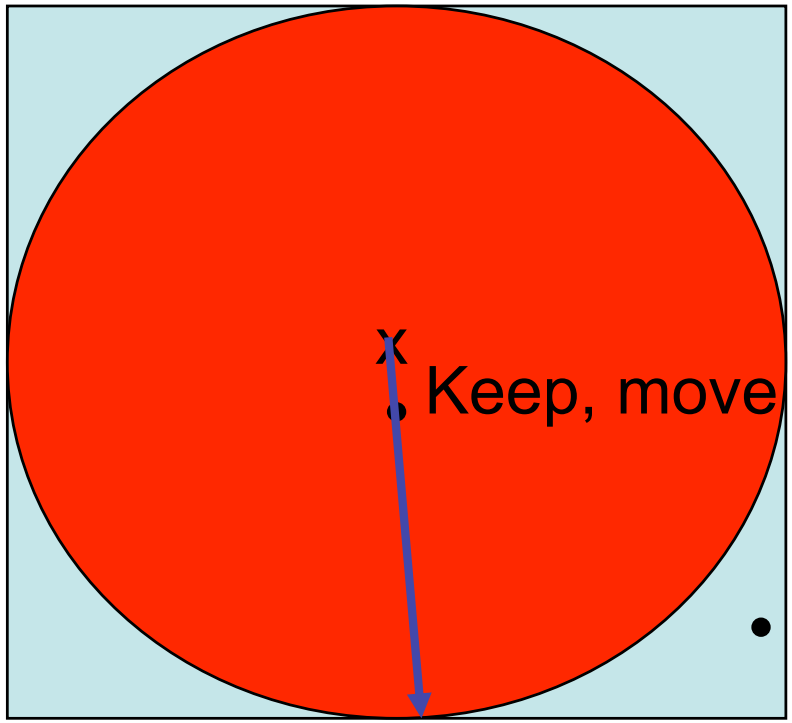
The rejection technique can be easily extended to several dimensions as indicated in the figure:




1. Generate a random vector *Y* uniformly distributed in Ω where Ω is a nice region, e.g., multidimensional rectangle, hypersphere, …
2. If *Y* is in *G*, accept
3. Else, repeat

# *Example*

Suppose we want to generate a random vector uniformly distributed on the surface of an n-dimensional unit sphere from a random vector generated in an n-dimensional hypercube.

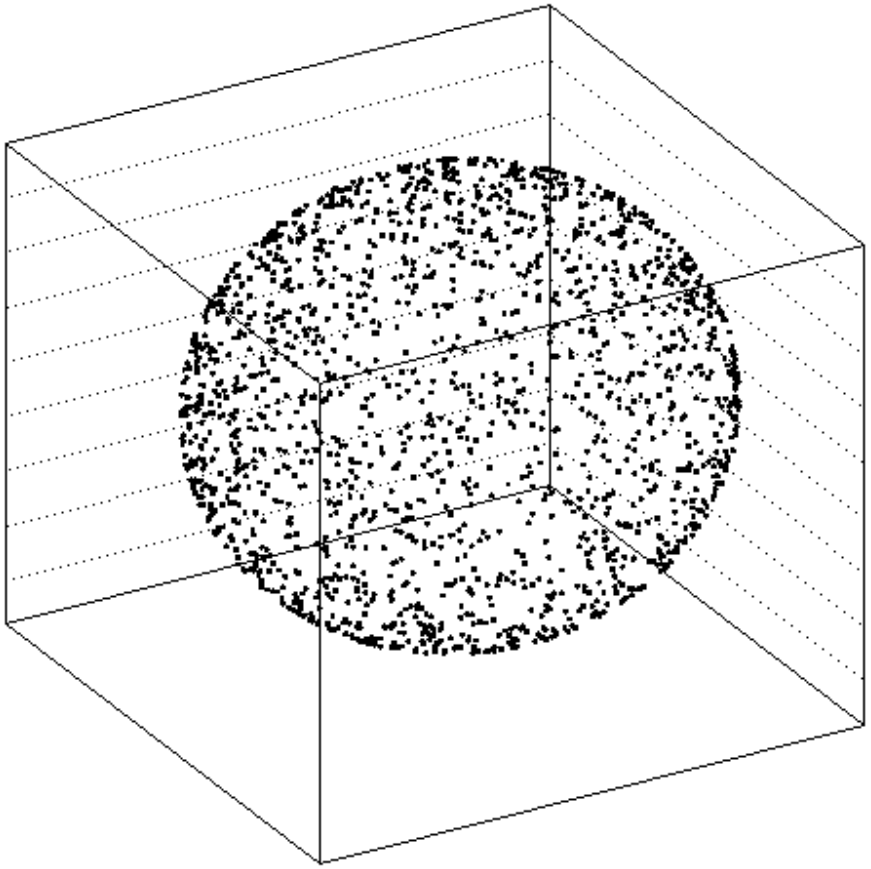



Check if point inside sphere. If yes, rescale coordinates to surface. Else, reject.

from R. Rubenstein, Simulation and the Monte Carlo Method

# *Spherical Surface*

```
*
* Try now the surface of a sphere (3D)
*
      Do I=1,10000,3
        x1=1-2*rvec(I)
        x2=1-2*rvec(I+1)
        x3=1-2*rvec(I+2)
        xsq=x1**2+x2**2+x3**2
        If (xsq.lt.1) then
          z1=x1/sqrt(xsq)
          z2=x2/sqrt(xsq)
          z3=x3/sqrt(xsq)
          Write(23,*) I,z1,z2,z3
        Endif
      Enddo
```



# *Simulation of Random Vectors*

Suppose we want to generate a random vector:

$$\vec{X} = \{X_1, \cdots, X_n\} \text{ with pdf } f(x_1, \cdots, x_n) \text{ and cdf } F(x_1, \cdots, x_n)$$

Case 1: the variables are independent

$$f(x_1, \cdots, x_n) = \prod_{i=1}^{n} f_i(x_i) \quad \text{where } f_i(x_i) = \int f(x_1, \cdots, x_n)\, d\vec{x}_{\text{except i}}$$

Can use the inverse transform method on each variable separately:

$$X_i = F_i^{-1}(U_i) \qquad i = 1, \cdots, n$$

# *Simulation of Random Vectors*

Case 2: Dependent variables:

$$F_1(X_1) = U_1$$
$$F_2(X_2 \mid X_1) = U_2$$
$$\vdots$$
$$F_n(X_n \mid X_1, \cdots, X_{n-1}) = U_n$$

Need to solve this system of equations. Efficiency typically depends on the order in which equations set up. More on this later.

# *Probability Distributions for Functions of Variables*

You have generated random numbers according to a pdf, $p_\lambda(\vec{\lambda})$, and want to find the probability distribution for a function of your variables:

$\vec{\lambda}$ according to $p_\lambda(\vec{\lambda})$ n dimensional vector

$g(\vec{\lambda}) : \Re^n \to \Re$ Function of your variables

$$p(g) = \int_{\Re^n} \delta\left(g - g(\vec{\lambda})\right) p(\vec{\lambda}) d\vec{\lambda} = \int_{g^{-1}(g)} \frac{p(\vec{\lambda})}{|\Delta g|} d\sigma(\vec{\lambda})$$

Integral over n-1 dimensional surface defined by $g(\vec{\lambda}) = g$

e.g., 1 parameter $p_g(g) = \dfrac{p_\lambda(\lambda_g)}{dg/d\lambda|_{\lambda_g}}$ $g(\lambda_g) = g$

# *Exercises*

1. Use the cumulative distribution function to generate rns following a Poisson distribution of mean $\nu$=0.2, $\nu$=20.

2. Generate a random variable from $p(x) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2} \qquad -R \le x \le R$

using the acceptance-rejection technique.

3. On the web page www.mpp.mpg.de/~caldwell/ss11/spectrum.dat , you will find the background spectrum measured in the GERDA test lab. Use the file to generate a background spectrum for 500000 events.

4. A radioactive source is located 1 cm above a detector surface, and the decay products are emitted isotropically. The decay products which hit the detector penetrate a distance s before stopping, with

$$p(s) \propto e^{-s/1\cdot 10^{-8}}$$

with s given in meters. Calculate & simulate the depth (normal to the surface) at which particles stop.

# *Monte Carlo Integration*

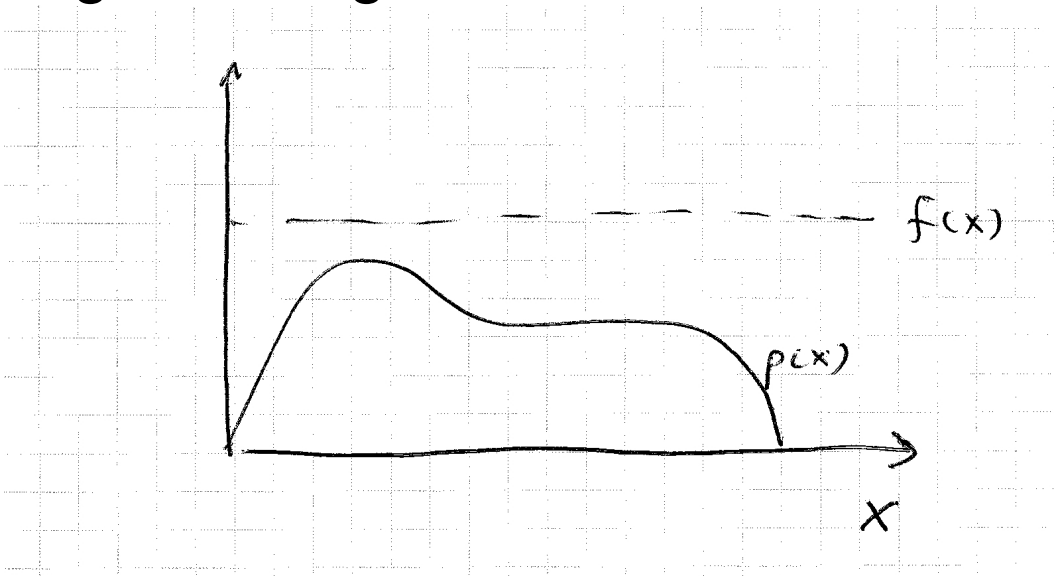
We now look at integration methods which are based on random numbers. There are many techniques which work well for low dimensional integrals with well-defined integration regions (e.g., Gaussian quadrature). In higher dimensions or complicated domains, these techniques do not work (well or at all). In these cases, Monte Carlo integration is the technique of choice.

As usual, there are several techniques available. We will start with the simplest, and then see how to improve the convergence properties (variance reduction techniques). Code words:

- Hit or Miss
- Sample mean
- Importance sampling, correlated sampling, stratified sampling
- Metropolis, Metropolis-Hastings, Markov Chain Monte Carlo

# *Hit or Miss Integration*

This method is very closely related to our Acceptance-Rejection technique for generating random numbers




Suppose we want the area under *p(x).* Then we can calculate the area under *f(x)* and multiply by the fraction of random numbers (generated uniformly under *f(x))* which are under *p(x).* If this fraction is *r*, then

$$I = \int_a^b p(x)\,dx = r\int_a^b f(x)\,dx$$

# *Hit or Miss Integration*

Note that each point in the plane has probability *r* to be under the curve *p(x).* We therefore have a Bernoulli process, and we can use the Binomial distribution to see how quickly the integration will converge. The steps are:

1. Pick a covering function *f(x)* which you can integrate.

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

2. Generate *N* points in the plane uniformly under *f(x).* Bayes' Theorem:

$$P(r \mid N,H) = \frac{P(H \mid N,r)P(r)}{\int_0^1 P(H \mid N,r)P(r)\,dr}$$

where *N* is the number of trials, *H* is the number of successes.

# *Hit or Miss Integration*

*P(r)* is our prior probability on the value of *r*. A smart guess will mean we need fewer points to converge (see example).

*P(H|N,r)* is the Binomial distribution:

$$P(H \mid N,r) = \frac{N!}{H!(N-H)!} r^H (1-r)^{N-H}$$

If we take *P(r)=const* (all values of *r* equally likely in range), then

$$P(r \mid H,N) = \frac{P(H \mid r,N)P(r)}{\int_0^1 P(H \mid r,N)P(r)dr} = \frac{\frac{N!}{(N-H)!H!} r^H (1-r)^{N-H}}{\int_0^1 \frac{N!}{(N-H)!H!} r^H (1-r)^{N-H} dr}$$

# *Hit or Miss Integration*

For the integration, we use $$\int_0^1 p^x (1-p)^{n-x}\, dp = \frac{x!(n-x)!}{(n+1)!}$$

The result is a β function, and for integer H,N reduces to

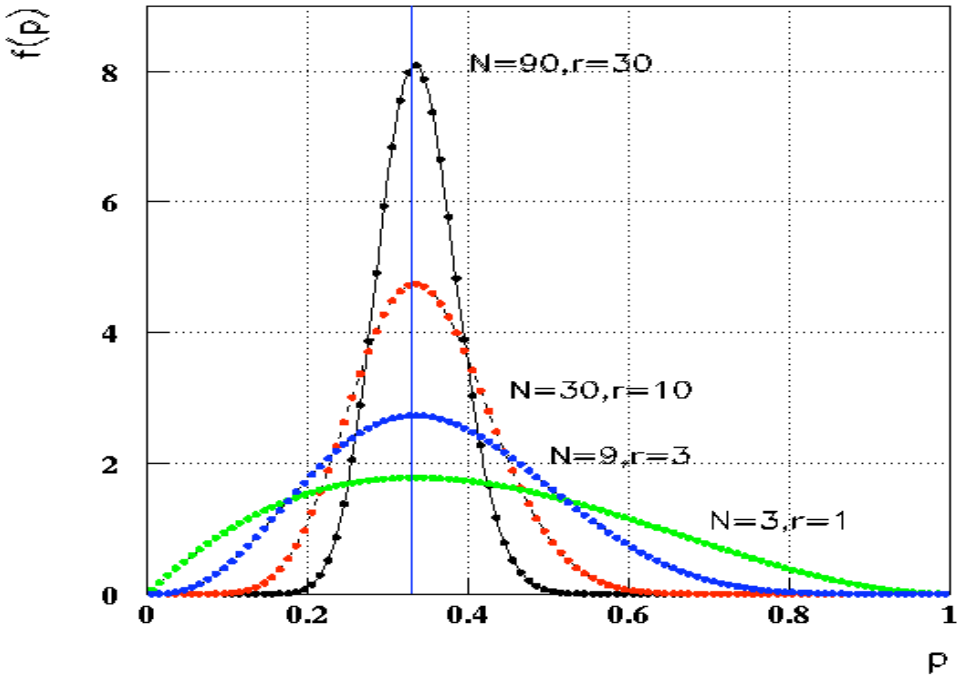
$$P(r \mid H,N) = \frac{(N+1)!}{H!(N-H)!} r^H (1-r)^{N-H}$$ Note maximum at *r=H/N*

The expectation value and variance are:

$$< r >= \int_0^1 \frac{(N+1)!}{H!(N-H)!} r^{H+1}(1-r)^{N-H}\, dr = \frac{(N+1)!}{H!(N-H)!}\frac{(H+1)!(N-H)!}{(N+2)!} = \frac{H+1}{N+2}$$

$$\sigma^2 = \frac{(H+1)(N-H+1)}{(N+3)(N+2)^2} =< r > (1- < r >)\frac{1}{N+3}$$

# *Hit or Miss Integration*




We are interested in large *N*. In this case,

$$\hat{r} \approx \frac{H}{N} \qquad \mathrm{var}(H) \approx N\hat{r}(1-\hat{r}) \qquad \mathrm{var}(r) \approx \frac{\hat{r}(1-\hat{r})}{N}$$

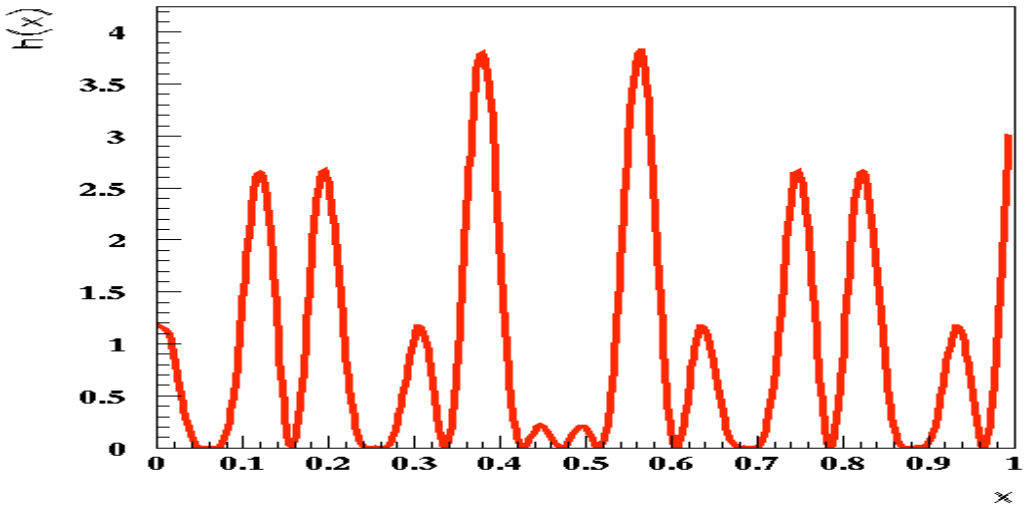
So width of *P(r)* decreases as $1/\sqrt{N}$

$\hat{I} = \hat{r}\Omega$ Uncertainty also scales as $1/\sqrt{N}$

# *Example*

Let's integrate the following function as an example:

$$h(x) = \left[\cos(50x) + \sin(20x)\right]^2 \quad 0 \le x \le 1$$




We can do this analytically, which is useful for studying how well our technique is working. The answer is:

$$\int \left(\left(\cos(50x) + \sin(20x)\right)^2\right) dx = \frac{1}{200}\left(\sin(100x) + 100x\right) - \frac{1}{210}\left(3\cos(70x) - 7\cos(30x)\right) - \frac{1}{80}\left(\sin(40x) - 40x\right) + C$$

http://www.teachers.ash.org.au/mikemath/resources/calculus.html

With our limits, this gives: ***I=0.9652***

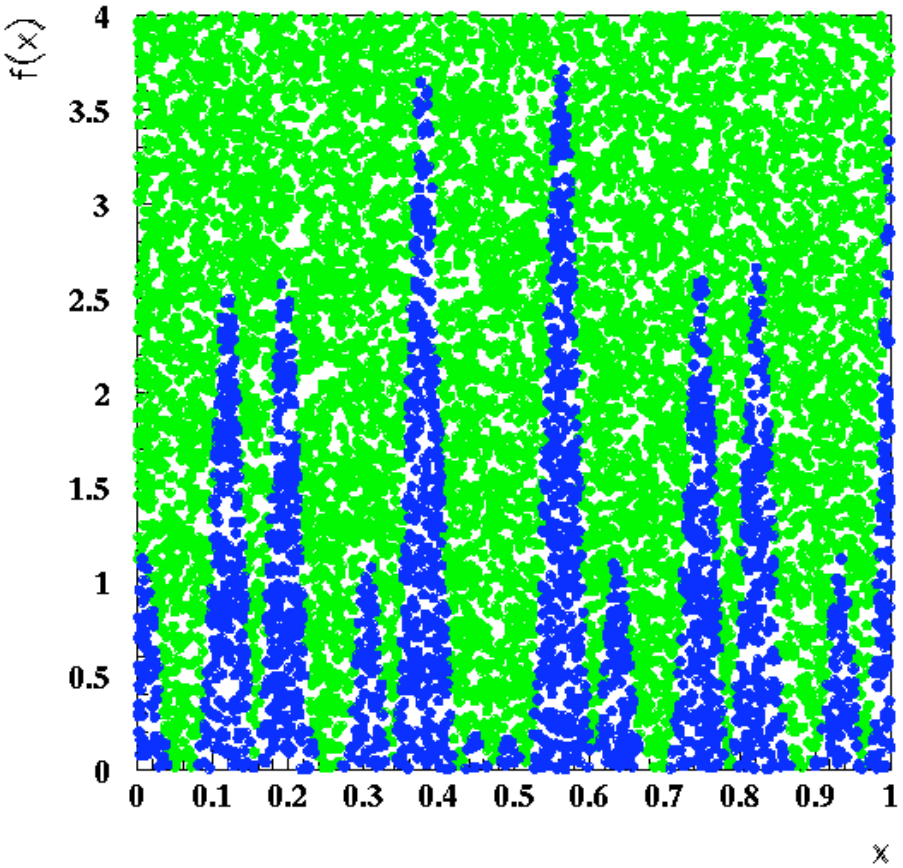
# *Example*

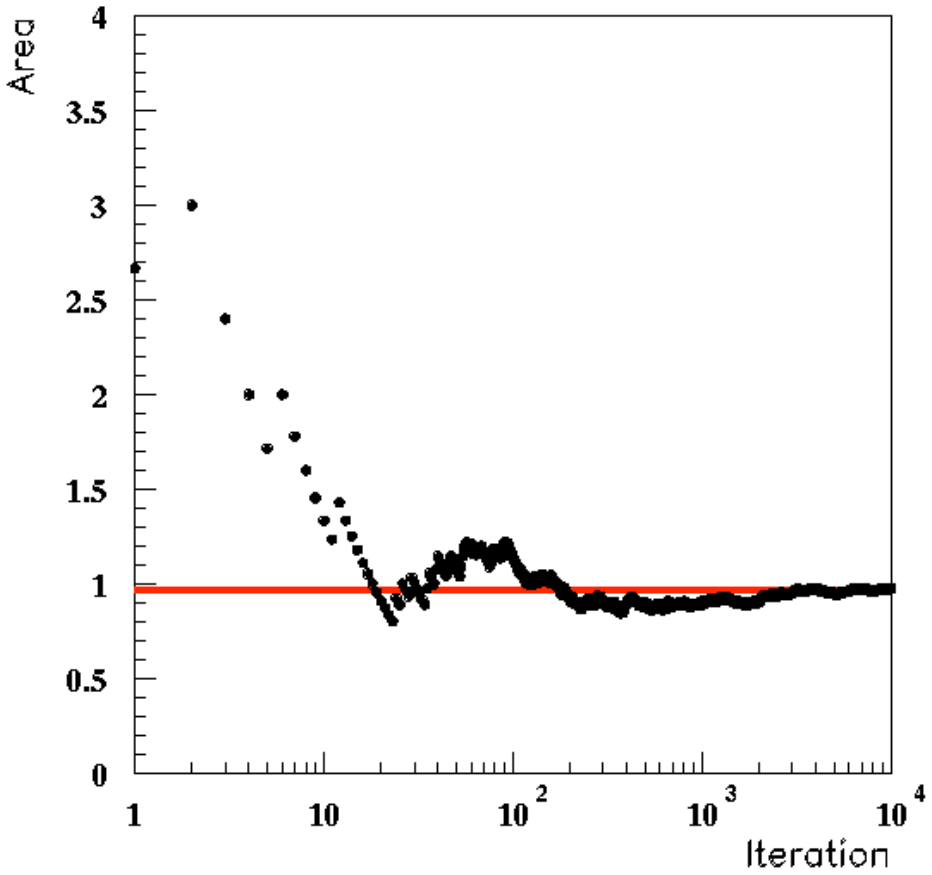
Here's the code:

```
* Generate some uniformly distributed random
* numbers
*
      Call Ranlux(rvec,10000)
      Call Ranlux(rvec1,10000)
*
      Igood=0
      Do I=1,10000
* the x coordinate is generated flat
         x=rvec(I)
         If (rvec1(I).le.testfun(x)/cover(x))
     &       Igood=Igood+1
         r=float(Igood+1)/float(I+2)
         area=r*4.
         error=area-0.9652
         errorest=sqrt(r*(1-r)/float(I+3))*4.
         write (25,*) I,x,r,area,error,errorest
*
      Enddo
```

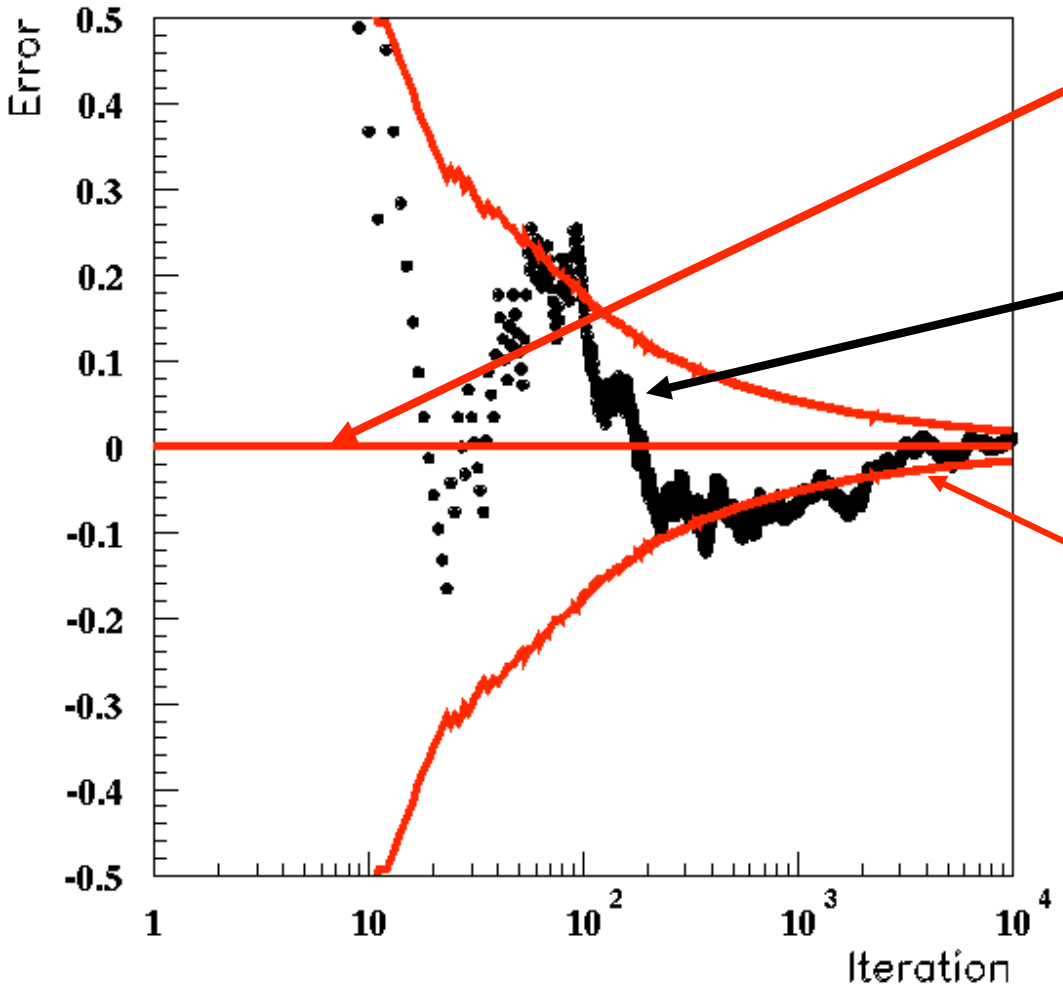
```
      Real Function testfun(x)
*
      Implicit None
      real x
*
      testfun=(cos(50*x)+sin(20*x))**2
*
      return
      end
*
      Real Function cover(x)
*
      Implicit None
      real x
*
      cover=4.
*
      return
      end
```

# *Example*


f(x)
4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
x


Area
4
3.5
3
2.5
2
1.5
1
0.5
0
1
10
$10^2$
$10^3$
$10^4$
Iteration

# *Example*




The known values

The calculated estimate

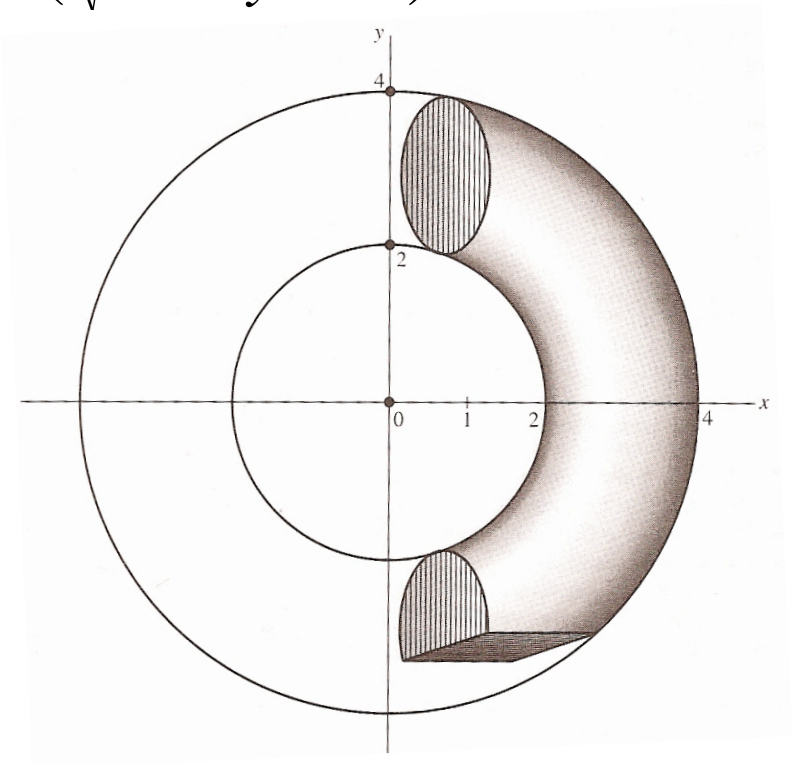
The calculated 1 sd uncertainty band

It is important that we can estimate the uncertainty !

# *Several Dimensions*

The hit-or-miss algorithm is easily extended to several dimensions, and complicated integration regions. Consider a region defined by the intersection of a torus with the edge of a box:

$$z^2 + (\sqrt{x^2 + y^2} - 3)^2 \le 1 \quad x \ge 1 \quad y \ge -3$$



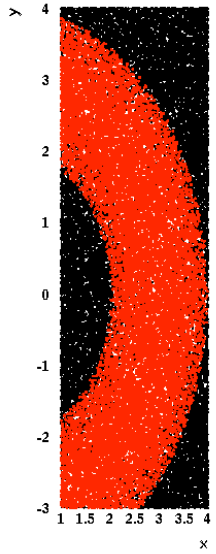

From Numerical Recipes in Fortran 77, 2nd Edition, W. Press et al.

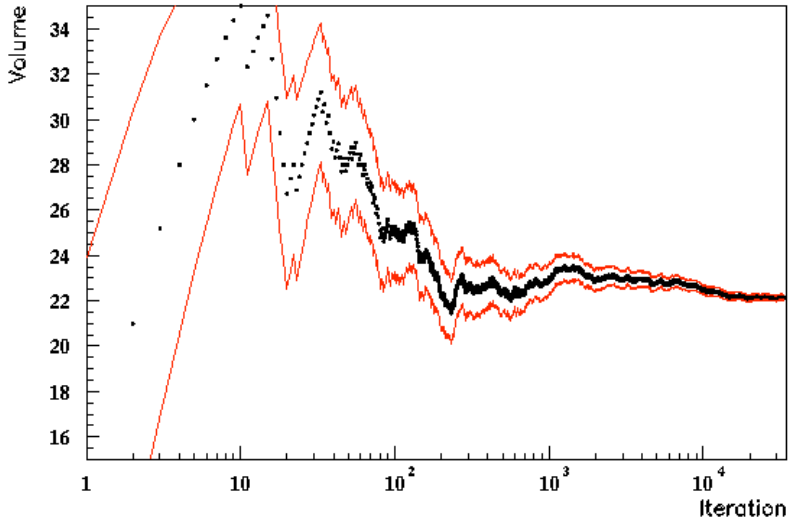
# *Example*

Suppose we want to evaluate the mass:

$$M = \int_{\Omega} \rho \, dxdydz$$

If the density is uniform, we can just calculate the volume of the object by simulating random numbers in a 3D box (1<x<4; -3<y<4; -1<z<1) and checking if the points are inside our volume.



# *Example*

Suppose the density $\rho$ is not a constant. How do we proceed ? We can supply a covering function for the density and draw r.n.'s in 4 dimensions… Let's look instead at the next method.

Sample-Mean Monte Carlo method. The basic idea here is to realize that an integral can be thought of as the expectation value of some random variable:

$$I = \int_a^b g(x)\,dx = \int_a^b \frac{g(x)}{f(x)} f(x)\,dx = E\left[\frac{g(x)}{f(x)}\right]$$

We require *f(x)>0 when g(x)≠0,* and *f(x)* is a pdf. The simplest case is to take

$$f(x) = \begin{cases} \dfrac{1}{b-a} & a < x < b \\ 0 & \text{otherwise} \end{cases}$$

# *Sample-Mean Monte Carlo*

Then $I = (b-a)E\left[g(x)\right]$

We can therefore estimate the integral as

$$\hat{I} = (b-a)\frac{1}{N}\sum_{i=1}^{N} g(x_i) \quad \text{where } x \text{ are generated according to } f(x) = \frac{1}{b-a}$$

In several dimensions, this becomes

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N}\frac{g(\vec{x}_i)}{f(\vec{x}_i)} \quad \text{where } \vec{x}_i \text{ are generated according to } f(\vec{x})$$

We can calculate the variance of the method as follows:

$$\text{var}(\hat{I}) = \text{var}\left((b-a)\frac{1}{N}\sum_{i=1}^{N} g(x_i)\right) = \left(\frac{b-a}{N}\right)^2 \text{var}\left(\sum_{i=1}^{N} g(x_i)\right)$$

# *Sample-Mean Monte Carlo*

Recall that we have: $\mathrm{var}(g(x)) = E[g(x)^2] - E[g(x)]^2$

where the expectation value has to be taken over the pdf, so

$$\mathrm{var}\left(\sum_{i=1}^{N} g(x_i)\right) = E\left[\left(\sum_{i=1}^{N} g(x_i)\right)^2\right] - E\left[\left(\sum_{i=1}^{N} g(x_i)\right)\right]^2$$

$$= E\left[Ng^2(x) + (N^2 - N)g(x_i)g(x_{j\neq i})\right] - E\left[Ng(x)\right]^2$$

$$= NE[g^2(x)] + (N^2 - N)E[g(x)]^2 - N^2 E[g(x)]^2$$

$$= N\left(E[g^2(x)] - E[g(x)]^2\right)$$

$$= N\left(\int_a^b g^2(x)\,\frac{1}{(b-a)}\,dx - \frac{I^2}{(b-a)^2}\right)$$

# *Sample-Mean Monte Carlo*

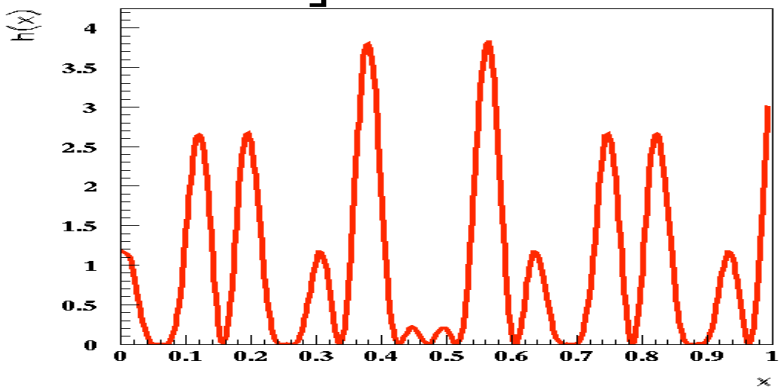
Putting the pieces together, we find:

$$\mathrm{var}(\hat{I}) = \left(\frac{b-a}{N}\right)^2 \mathrm{var}\left(\sum_{i=1}^{N} g(x_i)\right) = \left(\frac{b-a}{N}\right)^2 N\left(\int_a^b g^2(x)\,\frac{1}{(b-a)}\,dx - \frac{I^2}{(b-a)^2}\right)$$

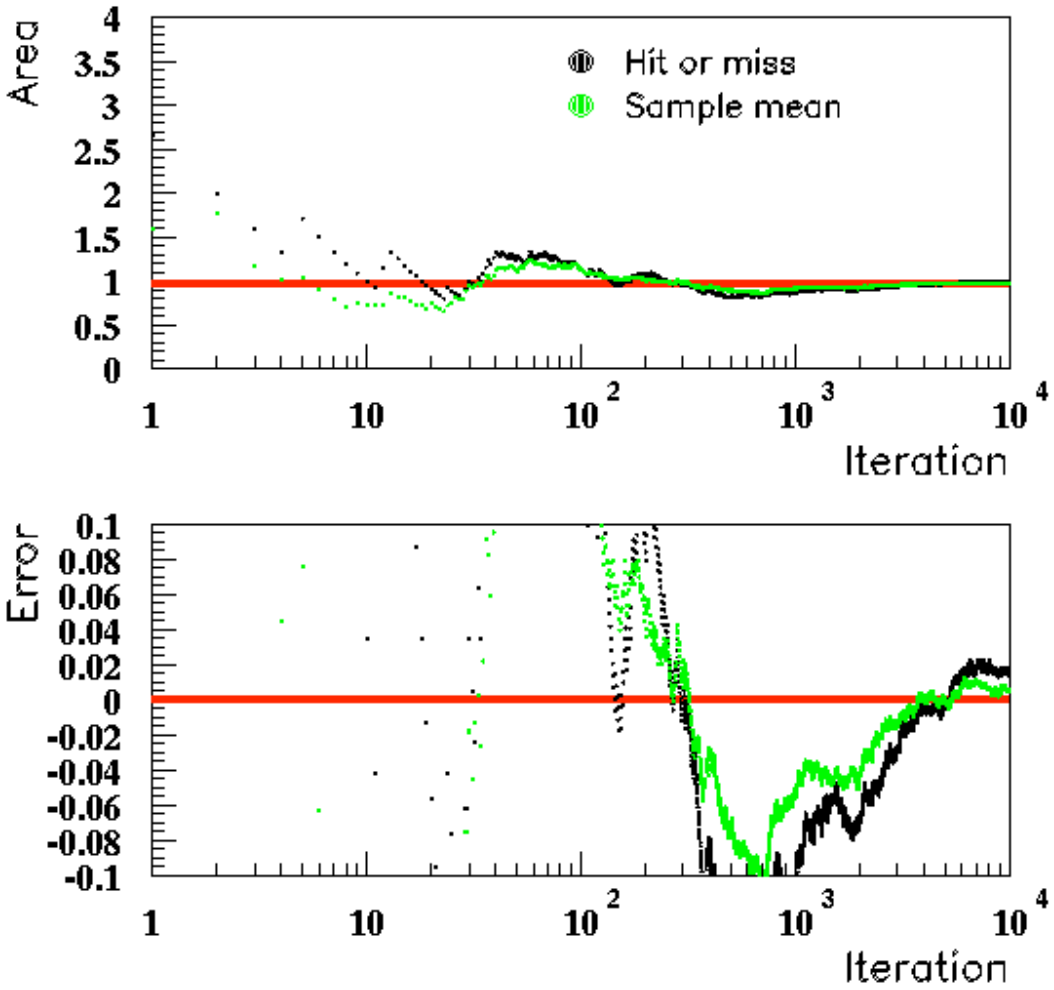
$$= \frac{1}{N}\left[(b-a)\int_a^b g^2(x)\,dx - I^2\right]$$

So again the variance decreases as *1/N*. Let's try it out on our previous function:

$$h(x) = \left[\cos(50x) + \sin(20x)\right]^2 \quad 0 \le x \le 1$$



# *Sample-Mean Monte Carlo*




Looks like the sample-mean method is converging more quickly. Let's see if we can understand why.

# *Efficiency of the Method*

We define the relative efficiency for two methods as follows:

$$\varepsilon = \frac{t_1 \operatorname{var}(\hat{I}_1)}{t_2 \operatorname{var}(\hat{I}_2)}$$

Let's compare the hit-or-miss and sample-mean methods:

$$\operatorname{var}(\hat{I}_1) = \Omega^2 \operatorname{var}(\hat{r}) = \Omega^2 \frac{E[r](1 - E[r])}{N} = \Omega^2 \frac{I/\Omega(1 - I/\Omega)}{N}$$

Hit-or-miss

$$= \frac{I(\Omega - I)}{N} = \frac{I(c(b-a) - I)}{N} = \frac{1}{N}\left[c(b-a)I - I^2\right]$$

$$\operatorname{var}(\hat{I}_2) = \frac{1}{N}\left[(b-a)\int_a^b g^2(x)\,dx - I^2\right]$$

Sample-mean

$$cI = \int_a^b cg(x)\,dx \geq \int_a^b g^2(x)\,dx \quad \text{because } c \geq g(x)$$

So sample-mean has smaller variance

# *Importance Sampling*

We look at techniques which have been developed to improve the MC integration. This means reducing the variance for a fixed number of iterations. Many techniques have been invented. We start with the most common.

In importance sampling, we concentrate the sampling in the region which contribute the most to the integral. We somehow need to use extra information to tell us where this 'most important region' is.

We start with the expression for the Sample-Mean method:

$$I = \int_a^b g(x)\,dx = \int_a^b \frac{g(x)}{f(x)} f(x)\,dx = E\left[\frac{g(x)}{f(x)}\right]$$

Except that now we will not take a flat *f(x)* but somehow choose it carefully so the larger values of *g(x)* are preferentially sampled.

# *Importance Sampling*

Our estimate for *I* is $\hat{I} = \frac{1}{N}\sum_{i=1}^{N}\frac{g(\vec{x}_i)}{f(\vec{x}_i)}$ for several dimensions

It is straightforward to show that the minimum variance is achieved when

$$f(\vec{x}) = \frac{|g(\vec{x})|}{\int |g(\vec{x})|\, d\vec{x}}$$

Or for $g(\vec{x}) > 0$ $f(\vec{x}) = \frac{g(\vec{x})}{\int g(\vec{x})\, d\vec{x}} = \frac{g(\vec{x})}{I}$

But this requires knowing *I*, which is what we want to find !

The message is, however, make *f* as close as possible to *g*

# *Example*

Let's go back to the integration over the truncated torus, and now assume that we have a density function which is a strong function of *z:*

$$\rho(z) = e^{5z}$$

In the sample-mean method, we would solve the integral as follows:

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N}\frac{g(\vec{x}_i)}{f(\vec{x}_i)} \quad \text{for several dimensions}$$

where we set: $f(x,y,z) = \frac{1}{\Delta x}\frac{1}{\Delta y}\frac{1}{\Delta z} = \frac{1}{3\cdot 7\cdot 2} = \frac{1}{42}$

and $$g(x,y,z) = \begin{cases} e^{5z} & z^2 + (\sqrt{x^2+y^2} - 3)^2 \le 1 \\ 0 & \text{otherwise} \end{cases}$$

# *Example*

We can estimate the uncertainty by numerically looking at how our estimate of the integral varies. For this, we consider subsamples of 100 iterations (as an example) and look at the variation of these estimates relative to the average of all.

$$\mathrm{var}[\hat{I}] \approx \frac{1}{N-1}\sum_{i=1}^{N}\left[\hat{I}_i - \hat{I}_{all}\right]^2$$

*N-1* 'degrees-of-freedom'

where

$$\hat{I}_j = \frac{1}{100}\sum_{i=1}^{100}\frac{g(\vec{x}_i)}{f(\vec{x}_i)}$$

we expect the distribution of the subsample means to be Gaussian distributed (Central Limit Theorem)

The uncertainty estimate on the integral would then be

$$\sigma_I \approx \sqrt{\frac{\mathrm{var}[\hat{I}]}{N}}$$

# *Example*

We will compare our 'Sample-Mean' estimate with an estimate where use use importance sampling. Let us try

$$f(x,y,z) = \frac{1}{3}\frac{1}{7}\left(\frac{4}{e^4 - e^{-4}} e^{4z}\right) \quad 1 < x < 4; -3 < y < 4; -1 < z < 1$$

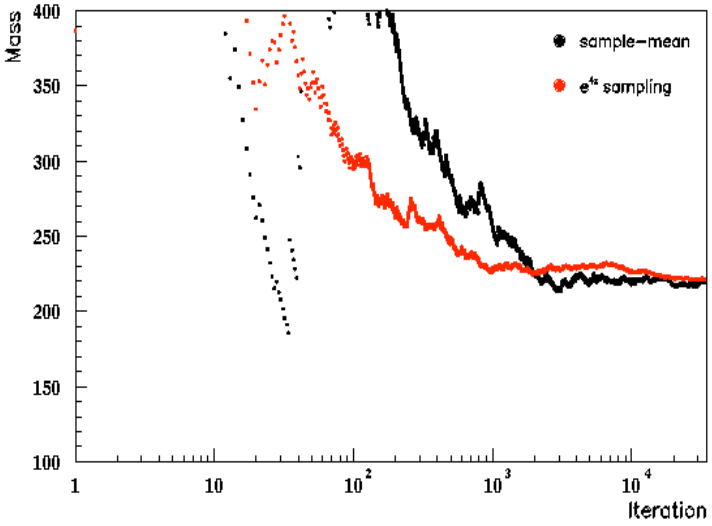
To get this from a uniform distribution, need

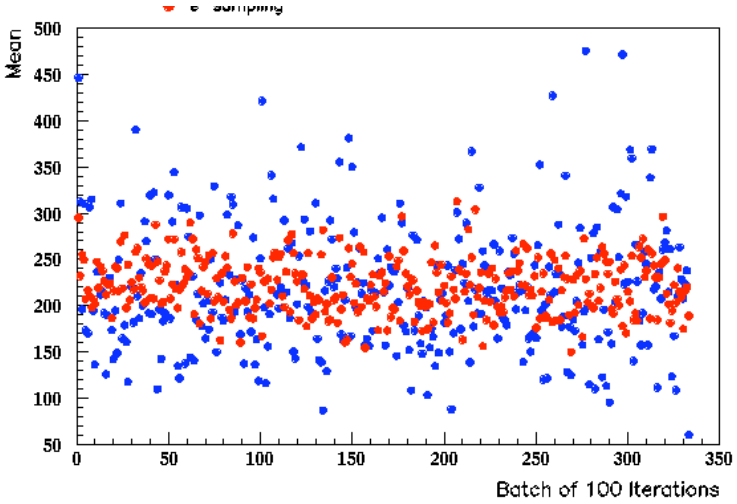
$$F(z) = \int_{-1}^{z} \left(\frac{4}{e^4 - e^{-4}} e^{4z'}\right) dz' = \frac{e^{4z} - e^{-4}}{e^4 - e^{-4}}$$

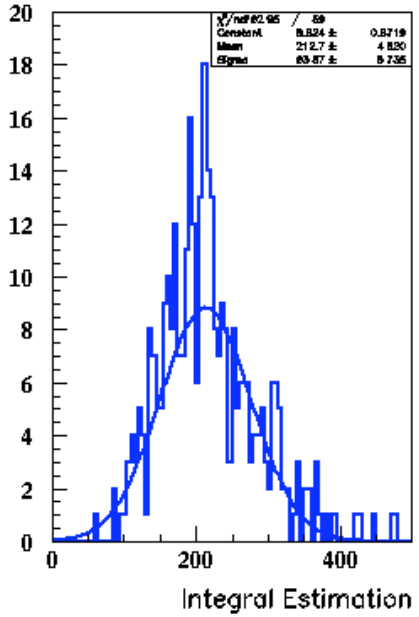
$$z(U) = F^{-1}(U) = \frac{1}{4}\ln\left[\left(e^4 - e^{-4}\right)U + e^{-4}\right]$$

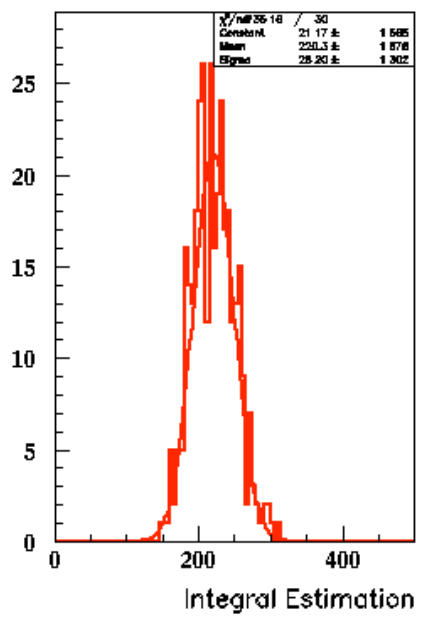
```
x=1.+3*rvec(I)
y=-3.+7*rvec(I+1)
z=alog(rvec(I+2)*(exp(4.)-exp(-4.))+exp(-4.))/4.
```

# *Example*




$$\sigma_I \approx \sqrt{\frac{\mathrm{var}[\hat{I}]}{N}}$$

# *Exercises*

1. Consider the following integral: $I = \int_0^{10} e^{-3x} \sin^3 x \; dx$

   a) Evaluate I using the hit-or-miss method, along with the error estimate. Set up your program to stop when the estimated error is less than 1%.
   b) Evaluate I using the Sample-Mean method, along with an error estimate.
   c) Evaluate I with an $e^{-3x}$ weighting. Compare the integral and estimated error to the previous two methods.

2. The LHC Collaborations say they want 5σ evidence for claiming the existence of the Higgs. Estimate a minimum number of background events which must be generated to be able to claim that the probability of a background fluctuation is less than this number.

# *Monte Carlo Optimization*

We want to solve problems of the sort

$$\max_{\theta \in \Theta} h(\theta)$$

Some numerical techniques for doing this (steepest descent, conjugate gradient, Newton-Raphson) work well in a small number of dimensions, but not in large dimensional spaces. They also require some analytic knowledge of the function to work well.

Here we consider Monte Carlo methods. First part of lecture follows: Monte Carlo Statistical Methods, C. Robert, G. Casella, 2nd Ed. Chapter 5.

# *Stochastic Exploration*

Brute force:

- generate values of $\Theta$ using a uniform distribution, and find the maximum using the approximation:

$$\max_{\theta \in \Theta} h(\theta) \approx h^* = \max(h(u_1), h(u_2), \ldots, h(u_m)) \quad u_i \sim U_\Theta$$

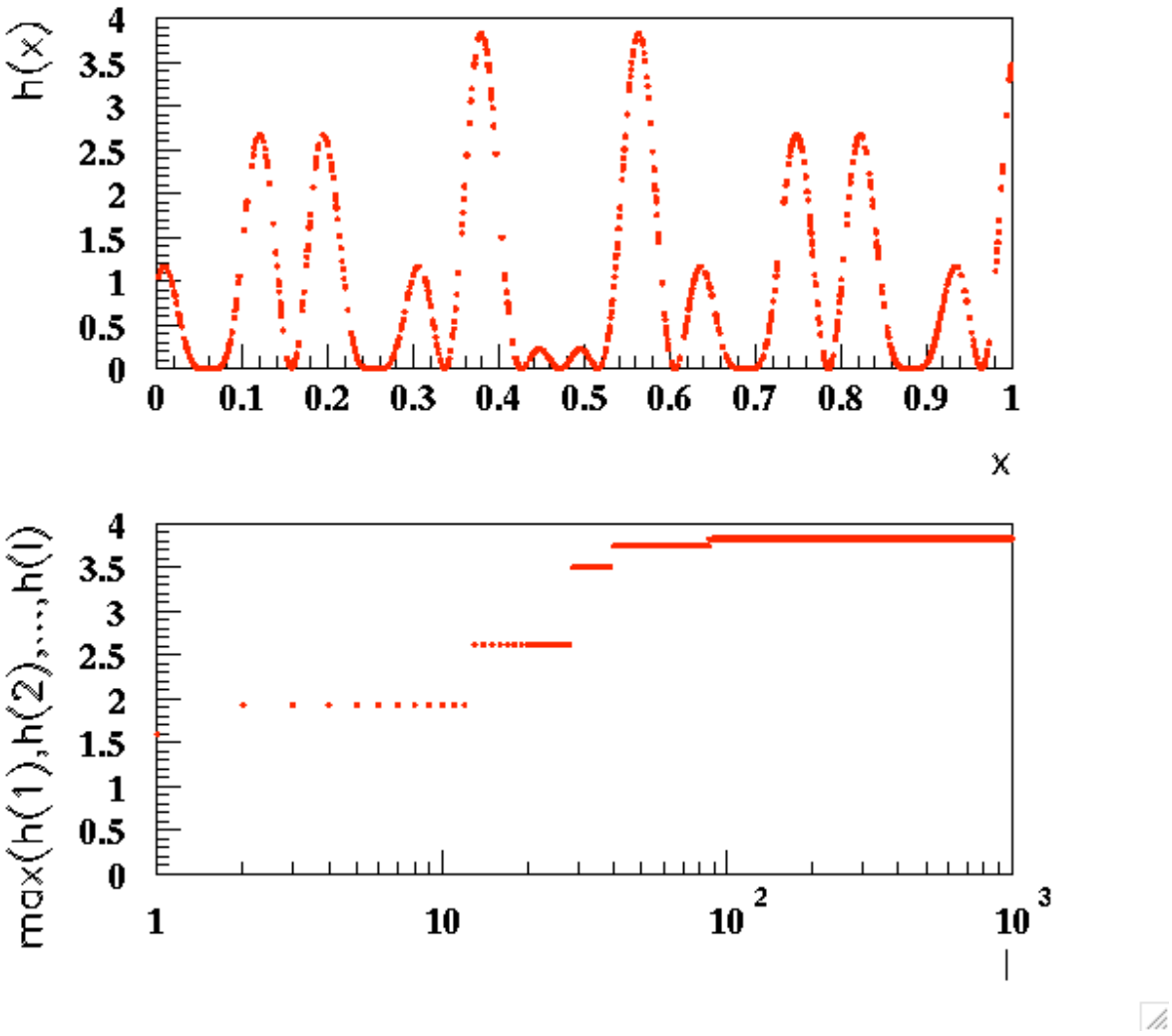
$$\text{If } h^* = h(u^*), \qquad \theta^* \approx u^*$$

This will always work, but it may be extremely slow. Obviously, if we can sample according to *h(θ)* we will be much more efficient.

Let's try it out on our old friend:

$$h(x) = \left[\cos(50x) + \sin(20x)\right]^2$$

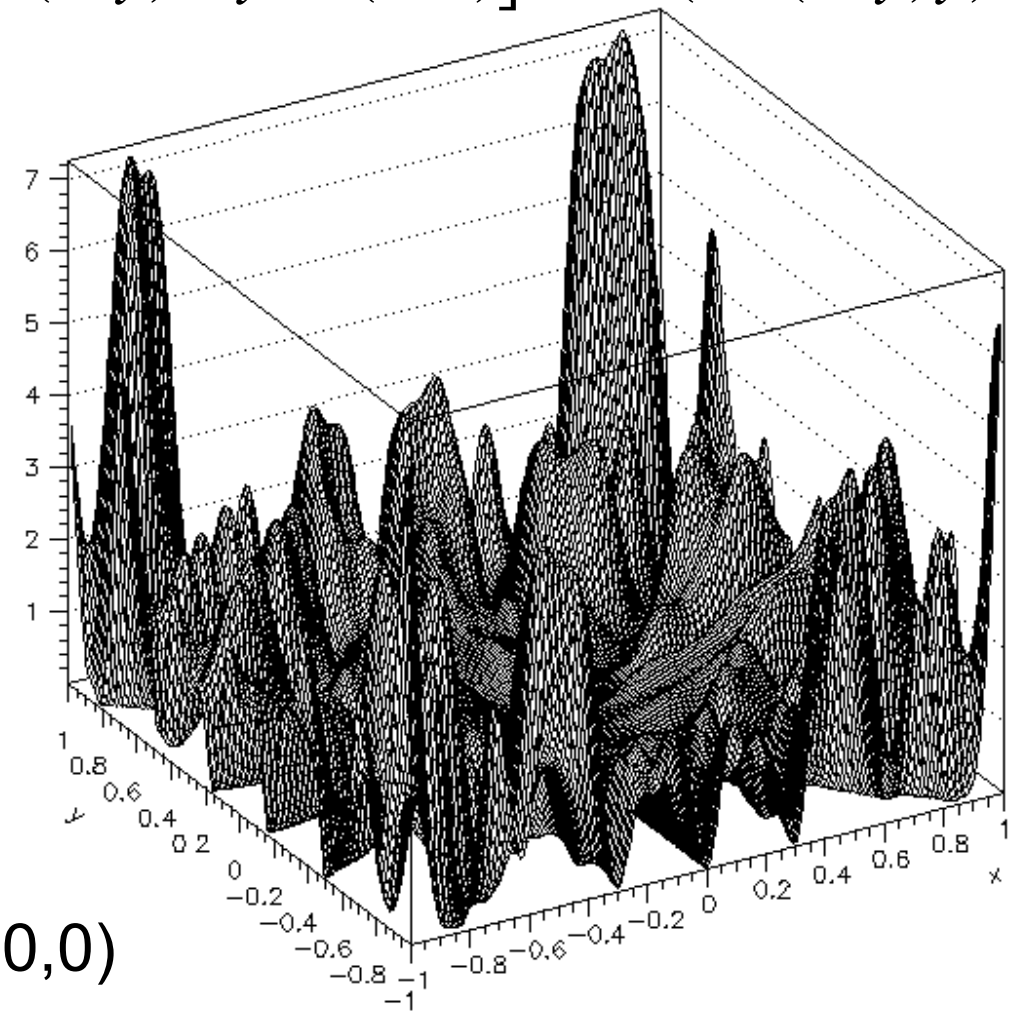
# *Stochastic Exploration*



# *Stochastic Exploration*

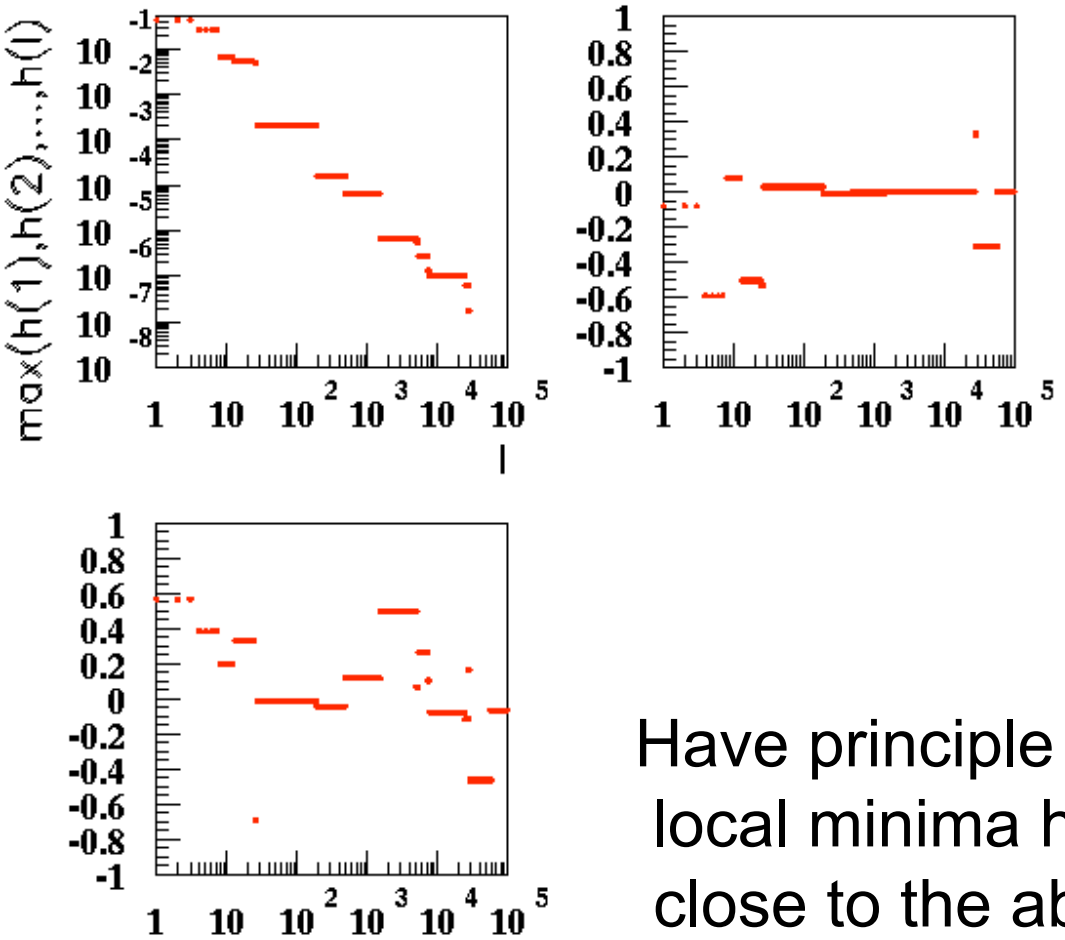
Let's look at a somewhat more complicated function:

$$h(x,y) = \left[x\sin(20y) + y\sin(20x)\right]^2 \cosh(\sin(10x)x) +$$
$$\left[x\cos(10y) - y\sin(10x)\right]^2 \cosh(\cos(20y)y)$$




Many local minima
Global minimum at (0,0)

# *Stochastic Exploration*

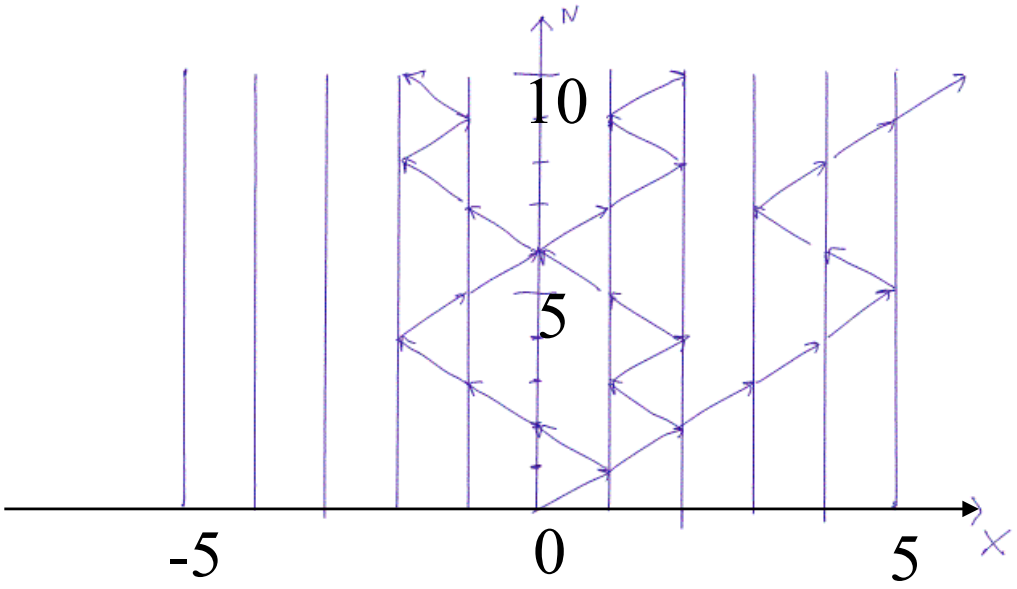



About 40000 iterations needed to reach real minimum. May not be stable.

Can we do better ?

Have principle problem that many of the local minima have minimum value very close to the absolute minimum.

# *Random Walk in 1-D*




Some possible paths

First question: what is the probability to be at x after n steps ?

Let $n_+$ represent the number of steps in the +x direction and $n_-$ the number of steps in the -x direction.

Note that n+x must be even

$$n_+ - n_- = x \qquad n_+ + n_- = n$$

so

$$n_+ = (n+x)/2 \qquad n_- = (n-x)/2$$

# *Random Walk in 1-D*

The distribution for $n_+$ is a Binomial distribution:

$$P(n_+) = \binom{n}{n_+} p^{n_+} q^{n-n_+} = \frac{n!}{n_+!(n-n_+)!} p^{n_+} q^{n-n_+}$$

so

$$< n_+ >= np \qquad < n_+^2 >= npq + n^2 p^2 \qquad \sigma^2_{n_+} = npq$$

Now for $x$

$$< x >=< 2n_+ - n >= 2 < n_+ > -n = n(2p-1)$$

$$< x^2 >=< (2n_+ - n)^2 >= 4 \ < n_+^2 > -4 < n_+ > n + n^2 = 4npq + n^2(1-4pq)$$

$$\sigma_x^2 = 4npq$$

# *Random Walk in 1-D*

We now look at the probability of returning to the origin. First, calculate the probability to be at *x* after *n* steps.

$$P_{0x}^{n} = \frac{n!}{\left((n+x)/2\right)!\left((n-x)/2\right)!} p^{(n+x)/2} q^{(n-x)/2}$$

$$2m = n \qquad P_{00}^{2m} = \frac{(2m)!}{m!m!} p^{m} q^{m}$$

For large n, use Stirling's approximation:

$$m! \approx m^{m+1/2} e^{-m} \sqrt{2\pi}$$

$$P_{00}^{2m} \approx \frac{(2m)^{2m+1/2} e^{-2m}}{m^{2m+1} e^{-2m} \sqrt{2\pi}} p^{m} q^{m} = \frac{2^{2m} (pq)^{m}}{\sqrt{m\pi}} = \frac{(4pq)^{m}}{\sqrt{m\pi}}$$

# *Random Walk in 1-D*

$$P_{00}^{2m} \approx \frac{(4pq)^m}{\sqrt{m\pi}}$$

Note that pq≤1/4, so that $P_{00}^{2m} \to 0$ for $m \to \infty$. The probability to be at the origin goes to 0. However, the number of returns to the origin after N steps

$$R(N) = \sum_{m=0}^{N/2} P_{00}^{2m} = \sum_{m=0}^{N/2} \frac{(2m)!}{m!m!} p^m q^m$$

Taking p=q=1/2,

$$R(N) = \sum_{m=0}^{N/2} \frac{(2m)!}{m!m!}(1/2)^{2m} = \frac{(N+1)!}{2^N (\frac{N}{2}!)^2} \xrightarrow[N\to\infty]{} \sqrt{\frac{2N}{\pi}}$$

Stirling's approx.

The state is recurrent - there is probability one of eventually returning to the origin. Only true of p=q=1/2.

# *Random Walk in 2,3-D*

In 1-D $$P_{00}^{2m} \propto \frac{1}{\sqrt{m\pi}}$$

In 2-D $$P_{00}^{2m} \propto \left(\frac{1}{\sqrt{m\pi}}\right)^2 \qquad \sum_{m=0}^{\infty} P_{00}^{2m} = \infty$$

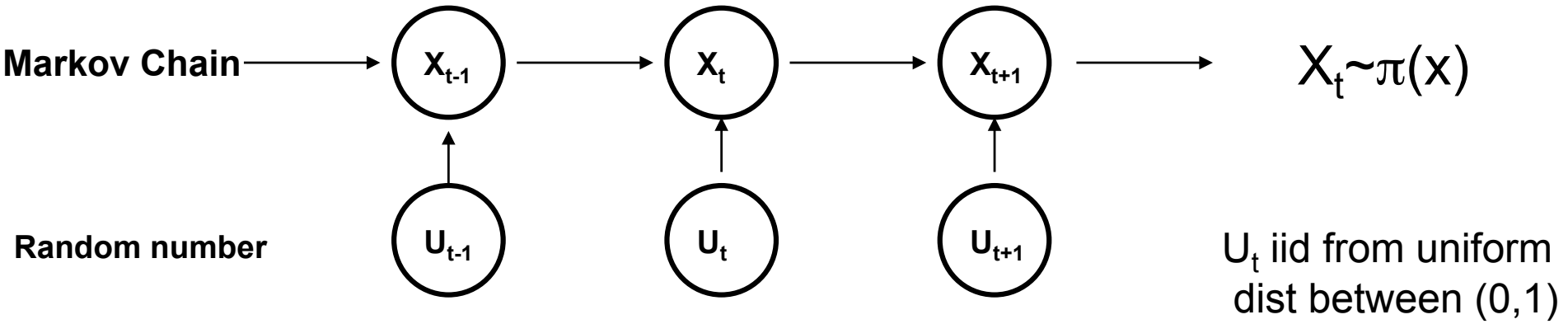
So the origin is recurrent

In 3-D $$P_{00}^{2m} \propto \left(\frac{1}{\sqrt{m\pi}}\right)^3 \qquad \sum_{m=0}^{\infty} P_{00}^{2m} < \infty$$

So the origin is not recurrent. There is a finite probability of never returning to the origin.

# *Markov Chains*

We set out the basic definitions and properties of Markov Chains, which underlie Markov Chain Monte Carlos (MCMC).




The $X_t$ are a sequence of random numbers, with limiting distribution $\pi(x)$, where $\pi(x)$ is the desired distribution to sample from. Often, have no good techniques available to perform this sampling.

Main feature of Markov Chains - can be easily generalized to large number of dimensions.

For a good review, see: First Course in Stochastic Processes, S. Karlin and H. Taylor, Academic Press

# *Markov Chains*

Basic Property of a Markov Process:

$$\Pr\{a < X_t \le b \mid X_{t_1} = x_1, \cdots, X_{t_n} = x_n\} = \Pr\{a < X_t \le b \mid X_{t_n} = x_n\}$$

$$t_1 < t_2 < \cdots < t_n < t$$

I.e., the probability distribution for the variable *X* depends only the current state, not on any previous behavior. For a finite or denumerable state space (which is always the case on a computer), have a Markov Chain. E.g., Poisson process is a continuous time Markov Chain.

# *Markov Chain Monte Carlo*

Basic Limit Theorem (for aperiodic, irreducible and recurrent Markov Chains)

$$\lim_{n\to\infty} P_{ii}^n = \frac{1}{\sum_{n=0}^{\infty} n f_{ii}^n} = \pi_i \qquad \lim_{n\to\infty} P_{ji}^n = P_{ii}^n = \pi_i$$

$\pi$ is the stationary distribution. Ergodic - does not depend on the starting point. Strongly ergodic class, all $\pi_i>0$.

Note that: $\lim_{n\to\infty} P_{jj}^n = \pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij}$ $\sum_{i=0}^{\infty} \pi_i = 1$ Eigenvalue equation

Detailed balance: $\pi_i P_{ij} = \pi_j P_{ji}$ Sufficient condition for $\pi_i$ to be stationary distribution of $P_{ij}$

# *Markov Chain Monte Carlo*

Proof: $\sum_{i=0}^{\infty} \pi_i P_{ij} = \sum_{i=0}^{\infty} \pi_j P_{ji}$ If have detailed balance

$$= \pi_j \sum_{i=0}^{\infty} P_{ji}$$

$$= \pi_j \quad \text{since} \quad \sum_{i=0}^{\infty} P_{ji} = 1$$ q.e.d.

So detailed balance is enough to prove stationarity.

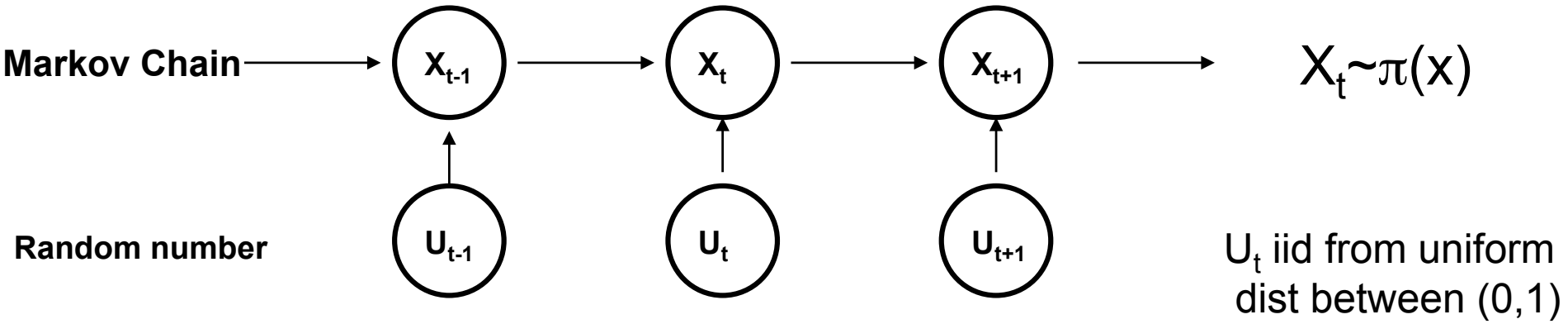
If we have an ergodic Markov chain with $\pi$~f, then

$$\lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} h(X_i) = E_f[h(X)]$$

Note that this is true even though the samples are not iid. Also, the convergence rate also scales as $O(1/\sqrt{N})$

# *Markov Chain Monte Carlo*

Goal of MCMC is to find a chain with $(\pi_i)_{i=0}^{\infty}$ =pdf of interest. Sampling according to the Markov Chain will then correspond to sampling from the desired pdf.




Define Markov Chain Monte Carlo as any method producing an ergodic Markov chain $X_t$ whose stationary distribution in the distribution of interest.

The original algorithm is due to Metropolis. Later generalized by Hastings.

# *Markov Chain Monte Carlo*

Uses:

1. Simulation of physical system which follows a known probability rule

$$x \sim \pi(x) \quad \text{where } x \text{ is a configuration}$$

2. Calculation of expectation values in a large number of dimensions

$$E[g(x)] = \int g(x)\pi(x)dx$$

3. Optimization with an annealing scheme

$$x^* = \arg\max \pi(x)$$

4. Learning (probability calculations)

# *Nicholas Metropolis*

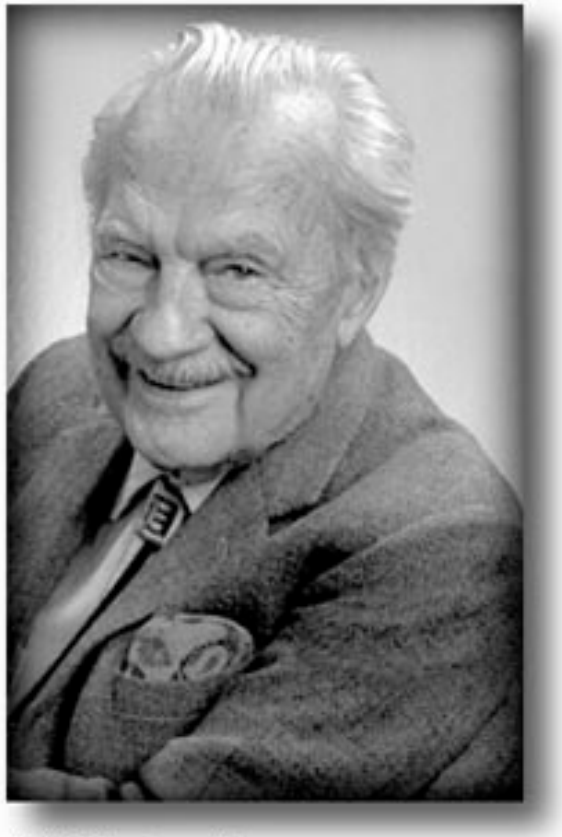
From Wikipedia, the free encyclopedia

Nicholas Constantine Metropolis (June 11, 1915 – October 17, 1999) was an American mathematician, physicist, and computer scientist.

Metropolis received his B.Sc. (1937) and Ph.D. (1941) degrees in experimental physics at the University of Chicago. Shortly afterwards, Robert Oppenheimer recruited him from Chicago, where he was at the time collaborating with Enrico Fermi and Edward Teller on the first nuclear reactors, to the Los Alamos National Laboratory. He arrived in the Los Alamos, on April 1943, as a member of the original staff of fifty scientists. After the World War II he returned to the faculty of the University of Chicago as an Assistant Professor. He came back to Los Alamos in 1948 to lead the group in the Theoretical (T) Division that designed and built the MANIAC I computer in 1952 and MANIAC II in 1957. (He chose the name MANIAC in the hope of stopping the rash of such acronyms for machine names, but may have, instead, only further stimulated such use.) From 1957 to 1965 he was Professor of Physics at the University of Chicago and was the founding Director of its Institute for Computer Research. In 1965 he returned to Los Alamos where he was made a Laboratory Senior Fellow in 1980.

# *Nicholas Metropolis*

Metropolis contributed several original ideas to mathematics and physics. Perhaps the most widely known is the Monte Carlo method. Also, in 1953 Metropolis co-authored the first paper on a technique that was central to the method known now as simulated annealing. He also developed an algorithm (the Metropolis algorithm or Metropolis -Hastings algorithm) for generating samples from the Boltzmann distribution, later generalized by W.K. Hastings.



Nick Metropolis

# *Metropolis Algorithm*

1. Suppose we have $X_t=x$. Generate a proposed new value, Y, according to a symmetric function $g(y,x)$. Symmetric means $g(y,x)=g(x,y)$.
2. Calculate $r=f(y)/f(x)$, where $f(x)$ is the desired density distribution. Generate a random number $U$ from a uniform distribution between 0,1. Then,

   set $X_{t+1}=y$ if $U<r$;

   else, $X_{t+1}=x$

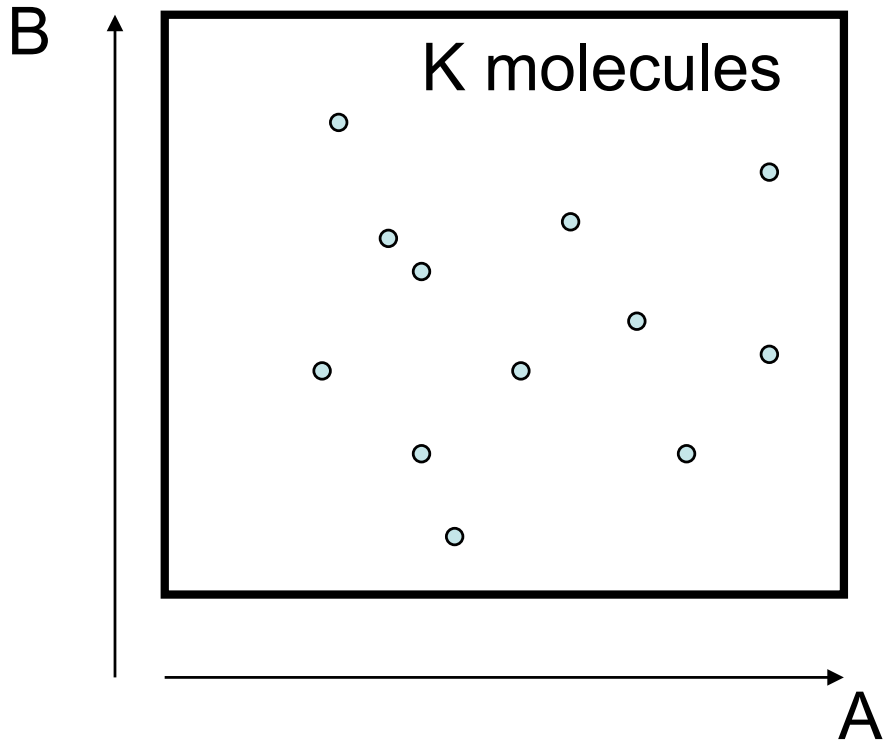
Note that all steps with $f(y)>f(x)$ are accepted. If $f(y)<f(x)$, take new position with probability r, else stay in current state.

Look at the example in the original Metropolis et al. paper:

N. Metropolis et al., J. Chem. Phys. 21 (1953) 1087.

# *Example*

Simple hard-sphere model for gas in 2-D. Spheres are placed initially on a regular lattice, then allowed to move. Evaluate the number of particles within a radius r from any particle.




Current state given by:

$$\{(x_i, y_i), i = 1, \ldots, K\}$$

Sphere diameter *d*

Target distribution uniform for all allowed configurations (non-overlapping balls, within boundaries)

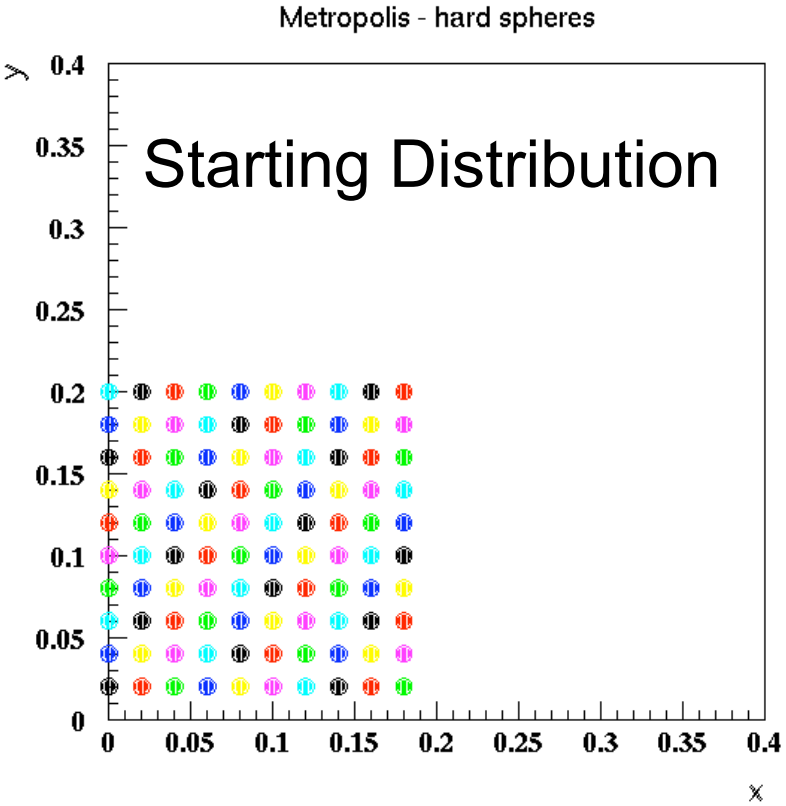
# *Example*

Algorithm:

1. Pick each particle in succession *(i)*
2. Perturb its position *($x_i \rightarrow x_i+\delta_1$, $y_i \rightarrow y_i+\delta_2$)* where the perturbations are taken from a uniform distribution from *[-s,s]*.
3. Apply periodic boundary conditions - if the particle is outside the square, it re-enters from the opposite side.
4. Calculate the change in energy of the system. If *$\Delta E<0$,* move is allowed and the particle is placed in the new position. If *$\Delta E>0$,* accept the move with probability *$\exp(-\Delta E/kT)$.*
5. Evaluate quantity of interest using sample mean

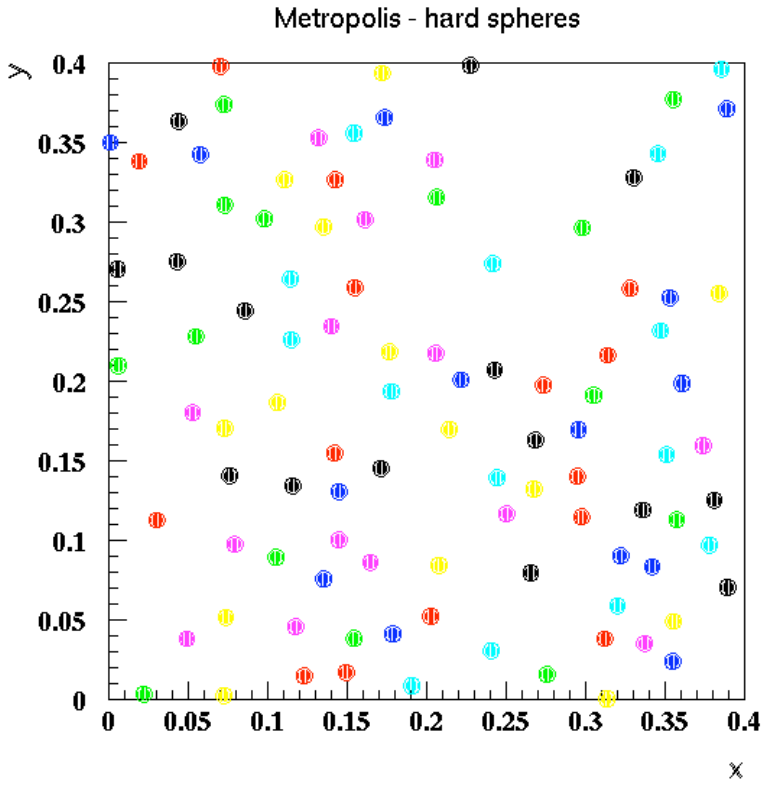
$$\overline{F} = \frac{\int F e^{-E/kT} d^{2N} p\, d^{2N} q}{\int e^{-E/kT} d^{2N} p\, d^{2N} q} \rightarrow \overline{F} \approx \frac{1}{M} \sum_{j=1}^{M} F_j$$

# *Example*



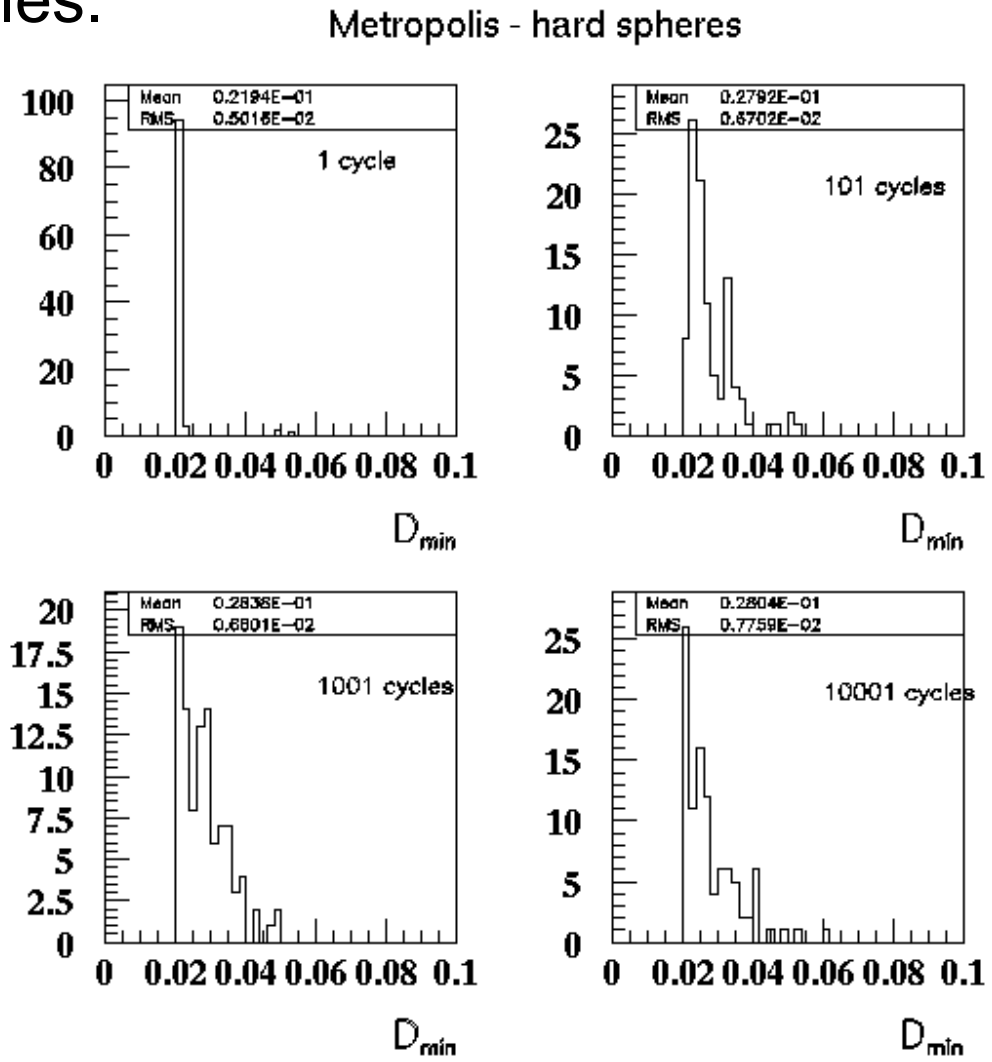

Parameters:
100 spheres
Box side 0.4
Sphere radius = 0.01
Maximum step size 0.02
(in each direction)

After 100 cycles



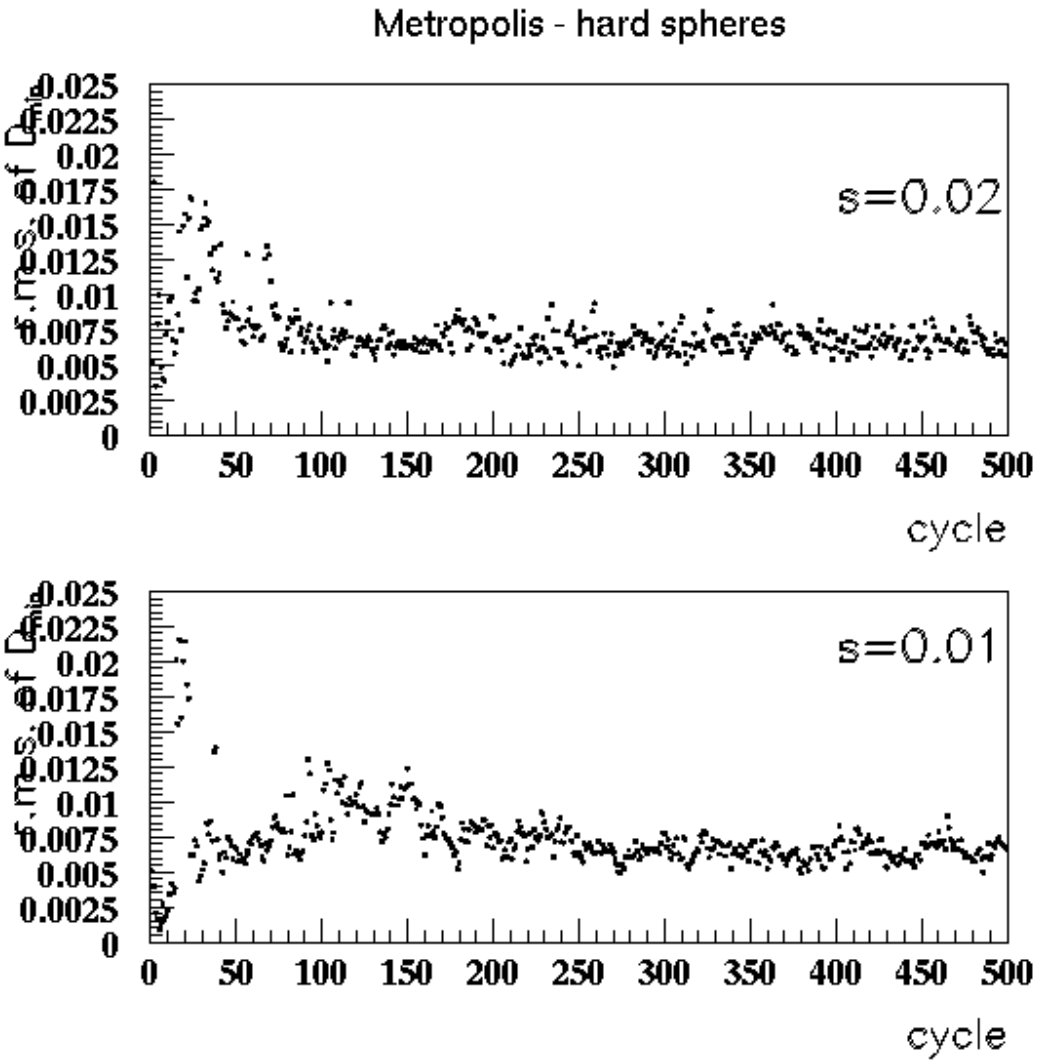
# *Example*

Let us see how quickly we reached an equilibrium situation. To study this, we look at the distribution of minimum separation for the nearest neighbor for all particles, and see how this distribution varies over cycles.



# *Example*

Here is the variation of the r.m.s. of those distributions versus the cycle number for different values of the maximum step size.



# *Metropolis-Hastings Algorithm*

The original algorithm is due to Metropolis. Later generalized by Hastings. Hastings showed that it is not necessary to use a symmetric proposal distribution, and proposed that the proposed new state can be generated from any $q(y|x)$.

Of course, the speed with which we reach the equilibrium distribution will depend on the choice of the proposal function.

# *Metropolis-Hastings Algorithm*

Given $x_i$:

1. Generate $Y_i$ according to $q(y|x_i)$
2. Take

$$X_{i+1} = \begin{cases} Y_i & \text{with probability } \rho(x_i, Y_i) \\ x_i & \text{with probability } 1 - \rho(x_i, Y_i) \end{cases}$$

where

$$\rho(x,y) = \min\left\{\frac{f(y)}{f(x)}\frac{q(x \mid y)}{q(y \mid x)}, 1\right\}$$

f is the *target* density and q is the instrumental or *proposal* distribution. Note that if $f(y_i)q(x_i \mid y_i) > f(x_i)q(y_i \mid x_i)$ then we always accept the new step. Else, it is accepted only with some probability. If the proposal distribution is symmetric, $q(y|x)=q(x|y)$ then probability only depends on ratio $f(y)/f(x)$.

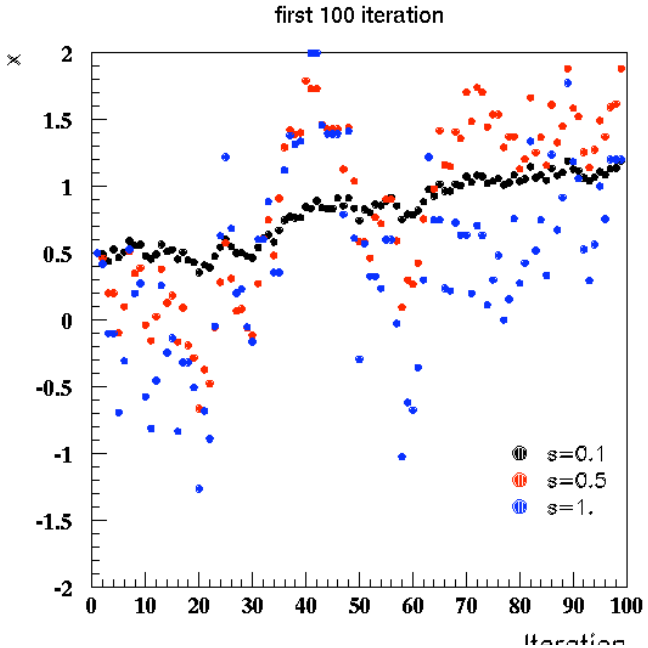
# *Example*

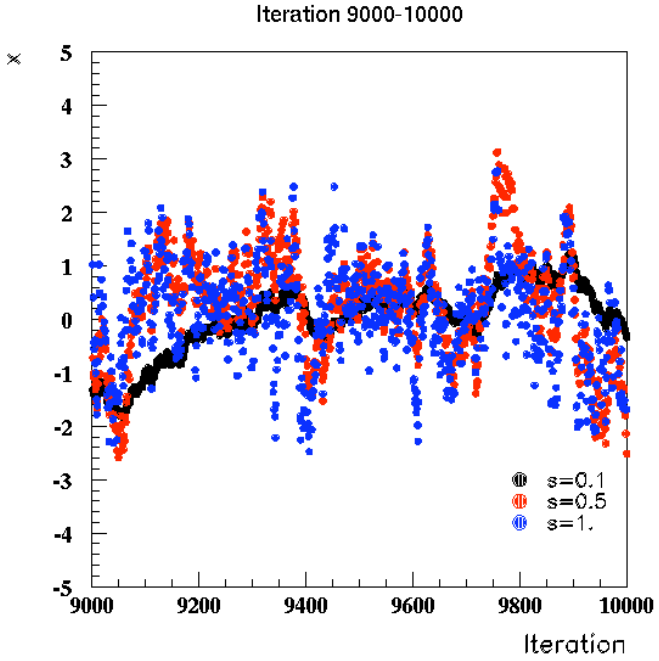
In this example, we look at the importance of the proposal distribution. Generate a Gaussian distribution with zero mean and $\sigma$=1 from a random walk Markov Chain with a step derived from a flat distribution as follows:

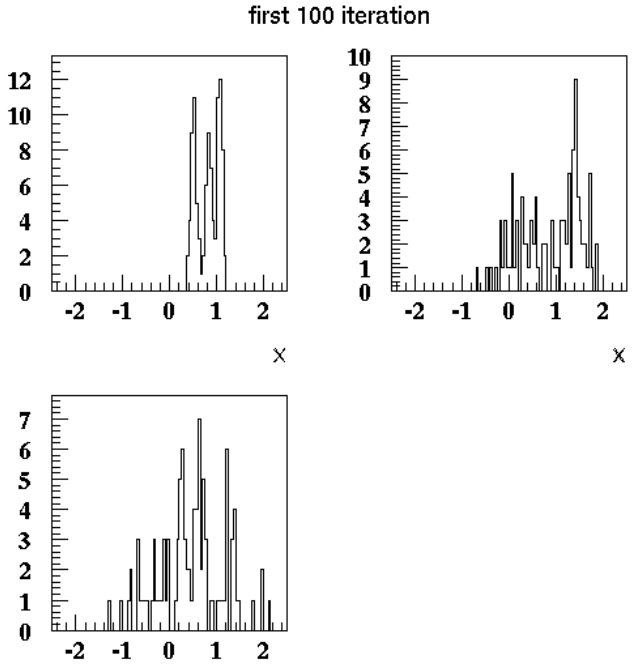
1. Generate a number from a flat distribution between *[-s,s]*; call it $\varepsilon$. Now set $y=x_t+\varepsilon$
2. Calculate $\rho = \min\left\{\frac{e^{-y^2/2}}{e^{-x^2/2}}, 1\right\}$ (note that *q(y|x)=q(x|y)*)
3. Set $x_{t+1}$=y if $U<\rho$, where *U* is a r.v. from a uniform distribution between (0,1)

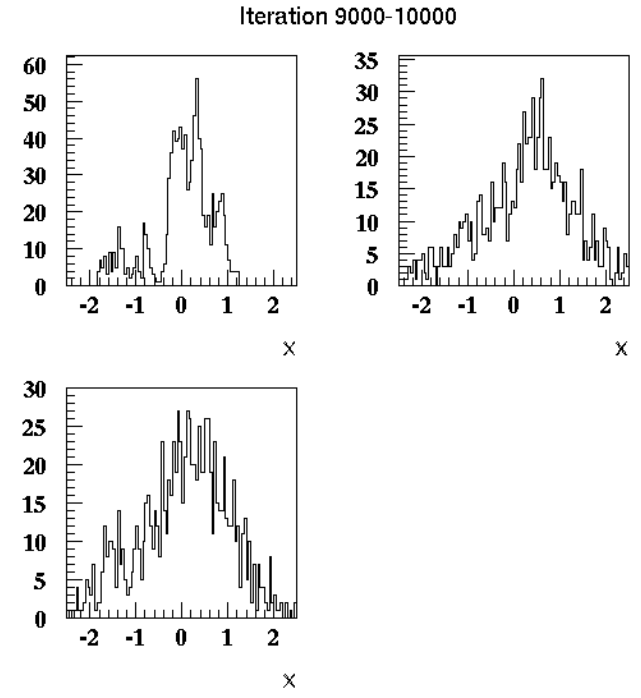
We will look to see how quickly we converge to the desired distribution depending on *s.*

# *Example*


first 100 iteration
x
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0 10 20 30 40 50 60 70 80 90 100
Iteration
s=0.1
s=0.5
s=1.


Iteration 9000-10000
x
5
4
3
2
1
0
-1
-2
-3
-4
-5
9000 9200 9400 9600 9800 10000
Iteration
s=0.1
s=0.5
s=1.


first 100 iteration
x
x


Iteration 9000-10000
x
x
x

# BAT → Software package for solving data analysis problems

## Code structured on Bayes' formula for parameter estimation

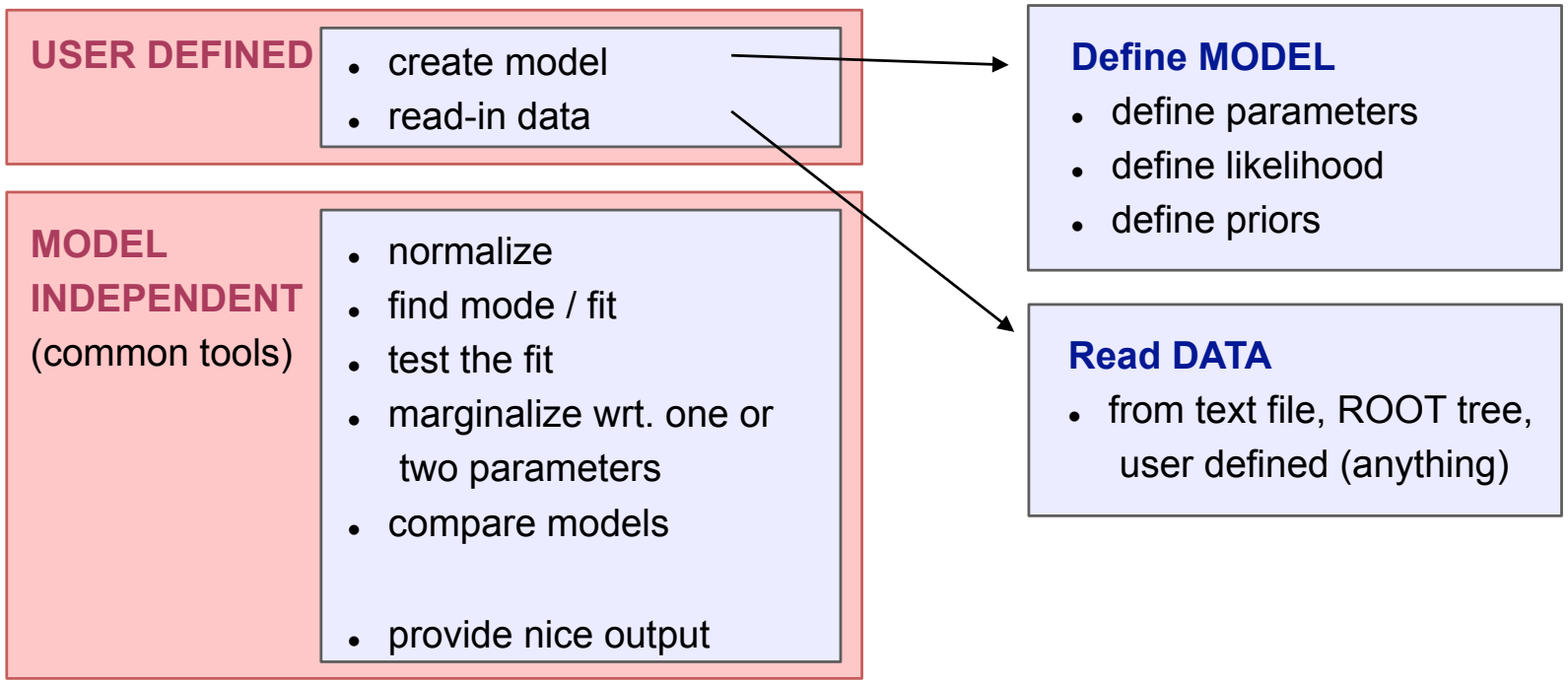
$$P(\vec{\lambda}, M|\vec{D}) = \frac{P(\vec{D}|\vec{\lambda}, M)P(\vec{\lambda}, M)}{P(\vec{D})}$$

- **The idea behind BAT**
- Merge common parts of every Bayesian analysis into a software package
- Provide flexible environment to phrase arbitrary problems
- Provide a set of well tested/tuned numerical algorithms and tools
- C++ based framework (flexible, modular)
- Interfaces to ROOT, Cuba, Minuit, user defined, ..
- can be downloaded from: http://www.mppmu.mpg.de/bat

# *The idea*

## Separate the common parts from the rest

- case specific: the model and the data
- common tools: all the rest



# *Markov Chain Monte Carlo (MCMC)*

- generally it is very difficult to obtain the full posterior PDF
  - number of parameters can be large
  - different input data will result in a different posterior
- also the visualization of the PDF in more than 3 dimensions is rather impractical and hard to understand
- usually one looks at marginalized posterior wrt. one, two or three parameters
  - a projection of the posterior onto one (two, three) parameter
  - integrating all the other parameters out
  - still numerically difficult

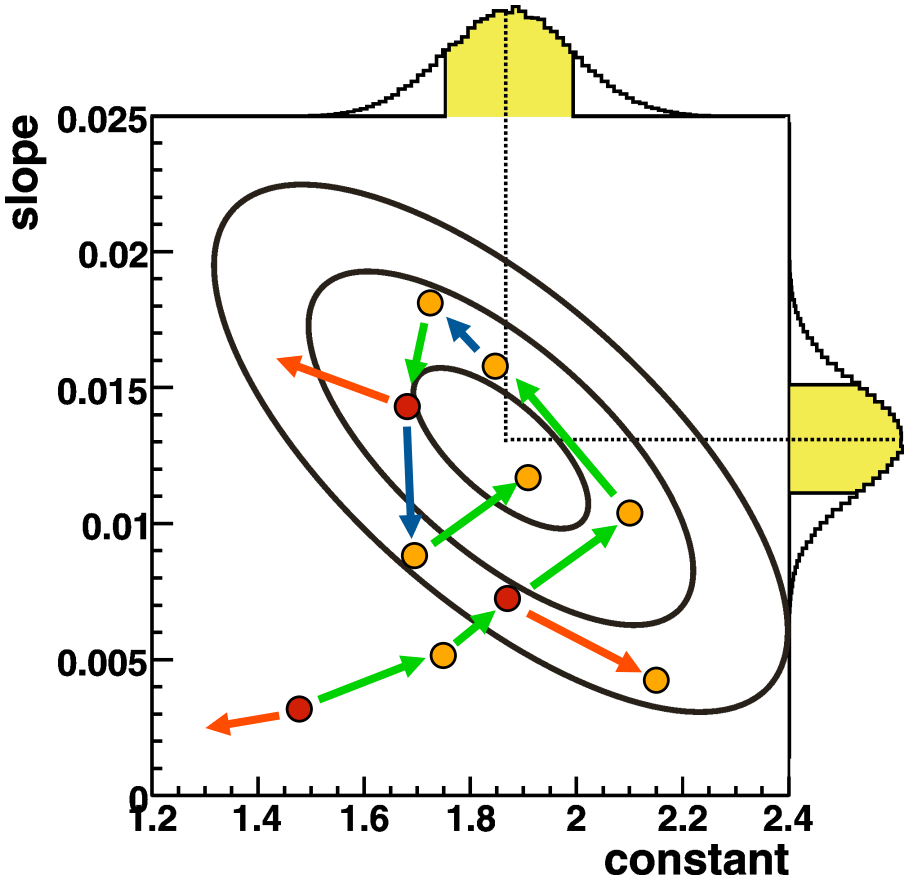
- the Markov Chain Monte Carlo revolutionized the area of Bayesian analysis

# *Scanning parameter space with MCMC*

- In Bayesian analysis use MCMC to scan parameter space of $\vec{\lambda}$
- MCMC converges towards underlying distribution
- Marginalize wrt. individual parameters while walking → obtain

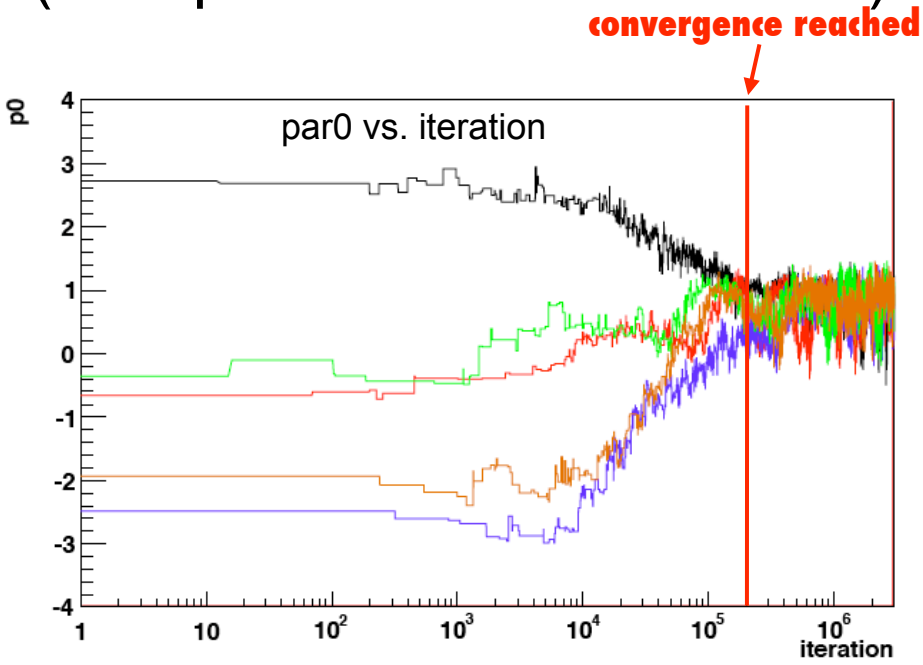
$$P(\lambda_i|\vec{D}) = \int P(\vec{\lambda}|\vec{D}) d\vec{\lambda}_{j\neq i}$$

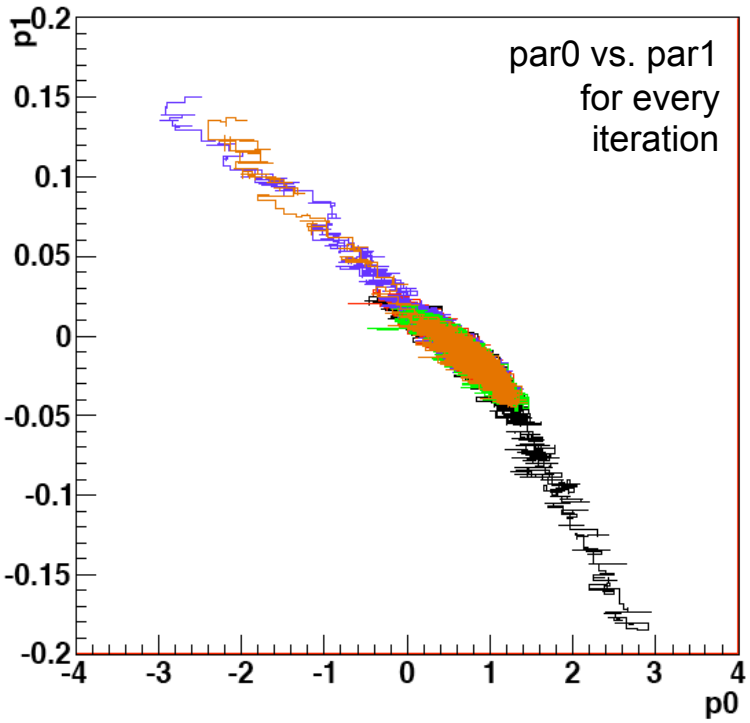
- Find maximum (mode)
- Uncertainty propagation



# *Analysis of Markov Chain*

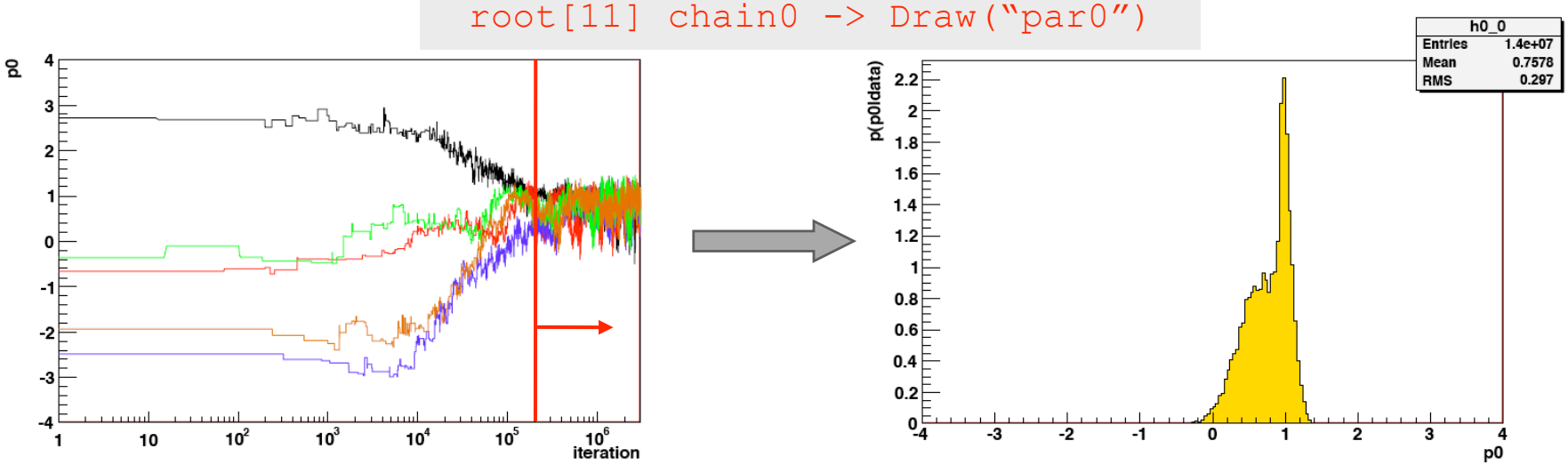
- the full chain(s) can be stored for further analysis and parameter tuning as ROOT TTree(s)
  - allows direct usage of standard ROOT tools for analysis
- Markov Chain contains the complete information about the posterior
  (except for the normalization)



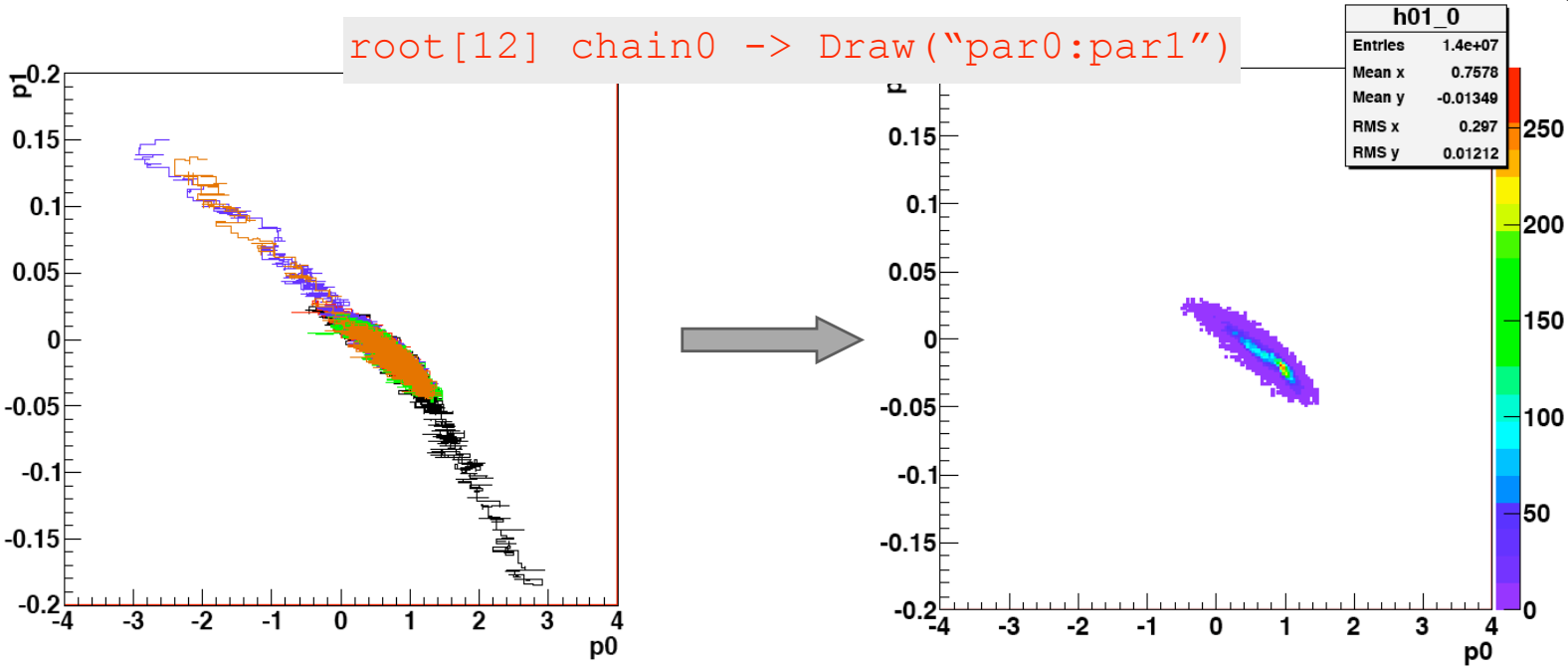



# *Obtaining marginalized distributions from TTree*

```
root[11] chain0 -> Draw("par0")
```




```
root[12] chain0 -> Draw("par0:par1")
```



# *Exercises*

1. Consider a random walk on the integers from -5,5 with transition probabilities:

$$P_{ij} = \begin{cases} \frac{1}{2} & \text{if } i = j, i = -4,...,4 \\ \frac{1}{4} & \text{if } j = i \pm 1, i = -4,...,4 \\ \frac{3}{4} & i = j = -5 \text{ or } 5 \\ \frac{1}{4} & i = -5, j = -4 \\ \frac{1}{4} & i = 5, j = 4 \\ 0 & \text{otherwise} \end{cases}$$

Find the invariant distribution. You can use a Markov Chain MC to give you a hint, and prove the final result mathematically. Try different starting points, and determine how many steps are needed before the sampling looks like sampling from the invariant distribution.

2. Write a Markov Chain Monte Carlo to produce numbers according to a Poisson distribution with mean 0.1, 20.

3. Optional - Perform the hard sphere simulation discussed in class for different choices of parameters. Once you have it working for hard spheres, think how you could modify the program to deal with 'soft-spheres'. Here you will need to use a potential energy function (find a reasonable one).