

## Part III: Monte Carlo Methods

### Topics:

- Introduction
- Random Number generators
- Special distributions
- General Techniques
- Multidimensional simulation

### References:

- *The Art of Computer Programming*, D.E. Knuth, Addison-Wesley, vol 2, 1969.
- *Monte Carlo Theory and Practice*, F. James, Rep. Prog. Phys., Vol. 43, 1980, 1145.
- *Portable Random Number Generators*, W.H. Press, S.A. Teukolsky, Computers in Physics, Vol. 6, No. 5, 1992, 522.

## Monte Carlo Techniques



Monte Carlo refers to any procedure that makes use of random numbers.

Monte Carlo methods are used in:

Simulation of natural phenomena  
Simulation of experimental apparatus  
Numerical analysis

### Random Numbers

What is a random number? Is 3?







No such thing as a single random number.

A sequence of random numbers is a set of numbers that have nothing to do with the other numbers in the sequence.

In a uniform distribution of random numbers in the range  $[0,1]$ , every number has the same chance of turning up.

Note that 0.00001 is just as likely as 0.50000

### How to generate a sequence of random numbers.

-  Use some chaotic system. (like balls in a barrel - Lotto 6-49).
-  Use a process that is inherently random:
  - radioactive decay
  - thermal noise
  - cosmic ray arrival
-  Tables of a few million truly random numbers do exist, but this isn't enough for most applications.
-  Hooking up a random machine to a computer is not a good idea. This would lead to irreproducible results, making debugging difficult.

## Random Number Generation

### Pseudo-Random Numbers

These are sequences of numbers generated by computer algorithms, usually in a uniform distribution in the range  $[0,1]$ .

To be precise, the algorithms generate integers between 0 and  $M$ , and return a real value:

$$x_n = I_n / M$$

An early example :

Middle Square (John Von Neumann, 1946)

To generate a sequence of 10 digit integers, start with one, and square it and then take the middle 10 digits from the answer as the next number in the sequence.

eg.  $5772156649^2 = 33317792380594909291$

so the next number is given by 

The sequence is not random, since each number is completely determined from the previous. But it appears to be random.

This algorithm has problems in that the sequence will have small numbers lumped together, 0 repeats itself, and it can get itself into short loops, for example:

$$6100^2=37210000$$

$$2100^2= 4410000$$

$$4100^2=16810000$$

$$8100^2=65610000$$

With more bits, long sequences are possible.  
38 bits → 750,000 numbers

A more complex algorithm does not necessarily lead to a better random sequence. It is better to use an algorithm that is well understood.

## Linear Congruential Method (Lehmer, 1948)

$$I_{n+1} = (a I_n + c) \bmod m$$

Starting value (seed) =  $I_0$

$a$ ,  $c$ , and  $m$  are specially chosen

$a, c \geq 0$  and  $m > I_0, a, c$

A poor choice for the constants can lead to very poor sequences.

example:  $a=c=I_0=7, m=10$

results in the sequence:  
7, 6, 9, 0, 7, 6, 9, 0,...

The choice  $c=0$  leads to a somewhat faster algorithm, and can also result in long sequences. The method with  $c=0$  is called: **Multiplicative congruential.**

## ★ Choice of modulus, $m$

$m$  should be as large as possible since the period can never be longer than  $m$ .

One usually chooses  $m$  to be near the largest integer that can be represented. On a 32 bit machine, that is  $2^{31} \approx 2 \times 10^9$ .

## ★ Choice of multiplier, $a$

It was proven by M. Greenberger in 1961 that the sequence will have period  $m$ , if and only if:

- i)  $c$  is relatively prime to  $m$ ;
- ii)  $a-1$  is a multiple of  $p$ , for every prime  $p$  dividing  $m$ ;
- iii)  $a-1$  is a multiple of 4, if  $m$  is a multiple of 4

With  $c=0$ , one cannot get the full period, but in order to get the maximum possible, the following should be satisfied:

- i)  $I_0$  is relatively prime to  $m$
- ii)  $a$  is a primitive element modulo  $m$

It is possible to obtain a period of length  $m-1$ , but usually the period is around  $m/4$ .



## RANDU generator

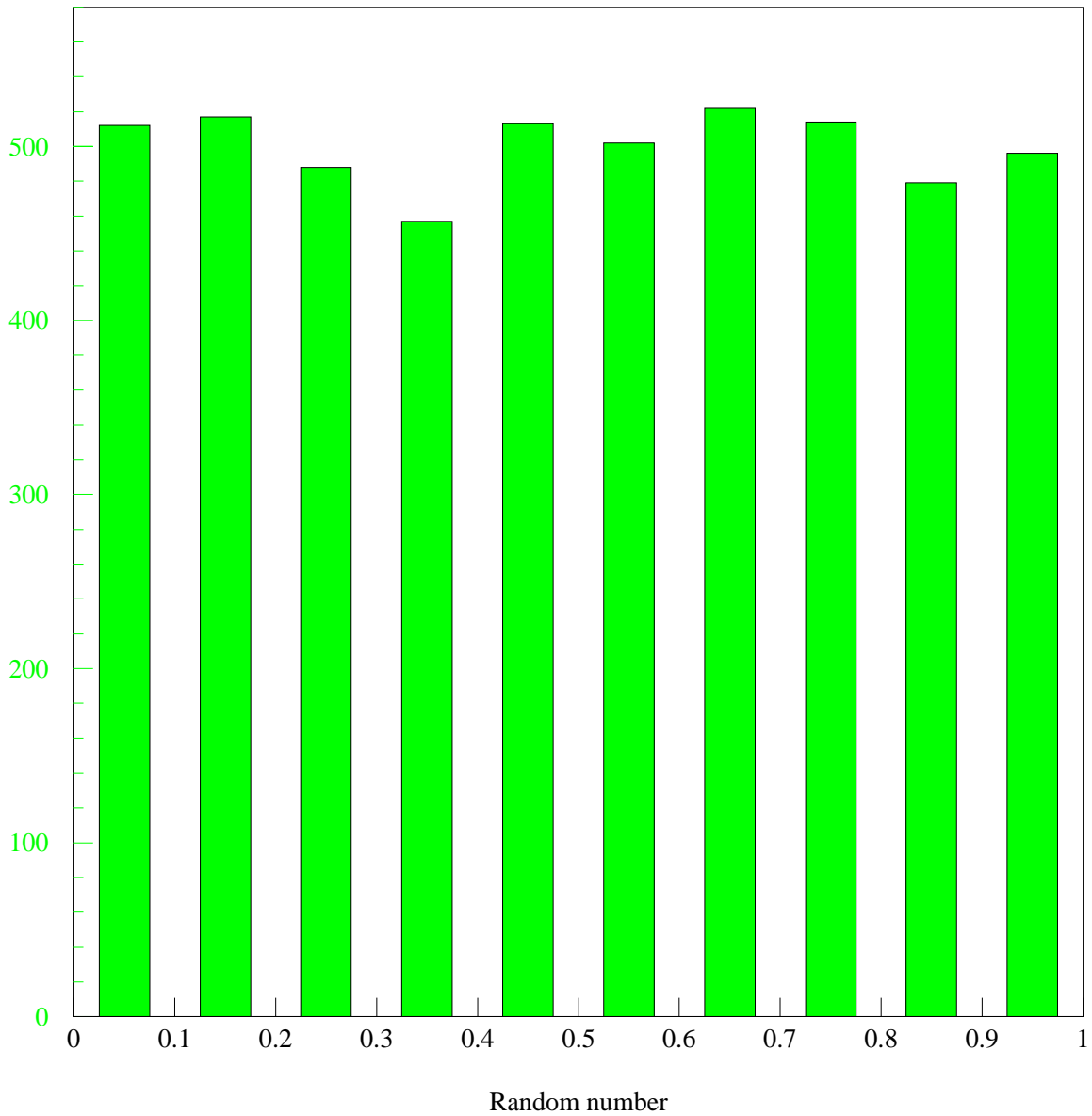
A popular random number generator was distributed by IBM in the 1960's with the algorithm:

$$I_{n+1} = (65539 \times I_n) \bmod 2^{31}$$

This generator was later found to have a serious problem...

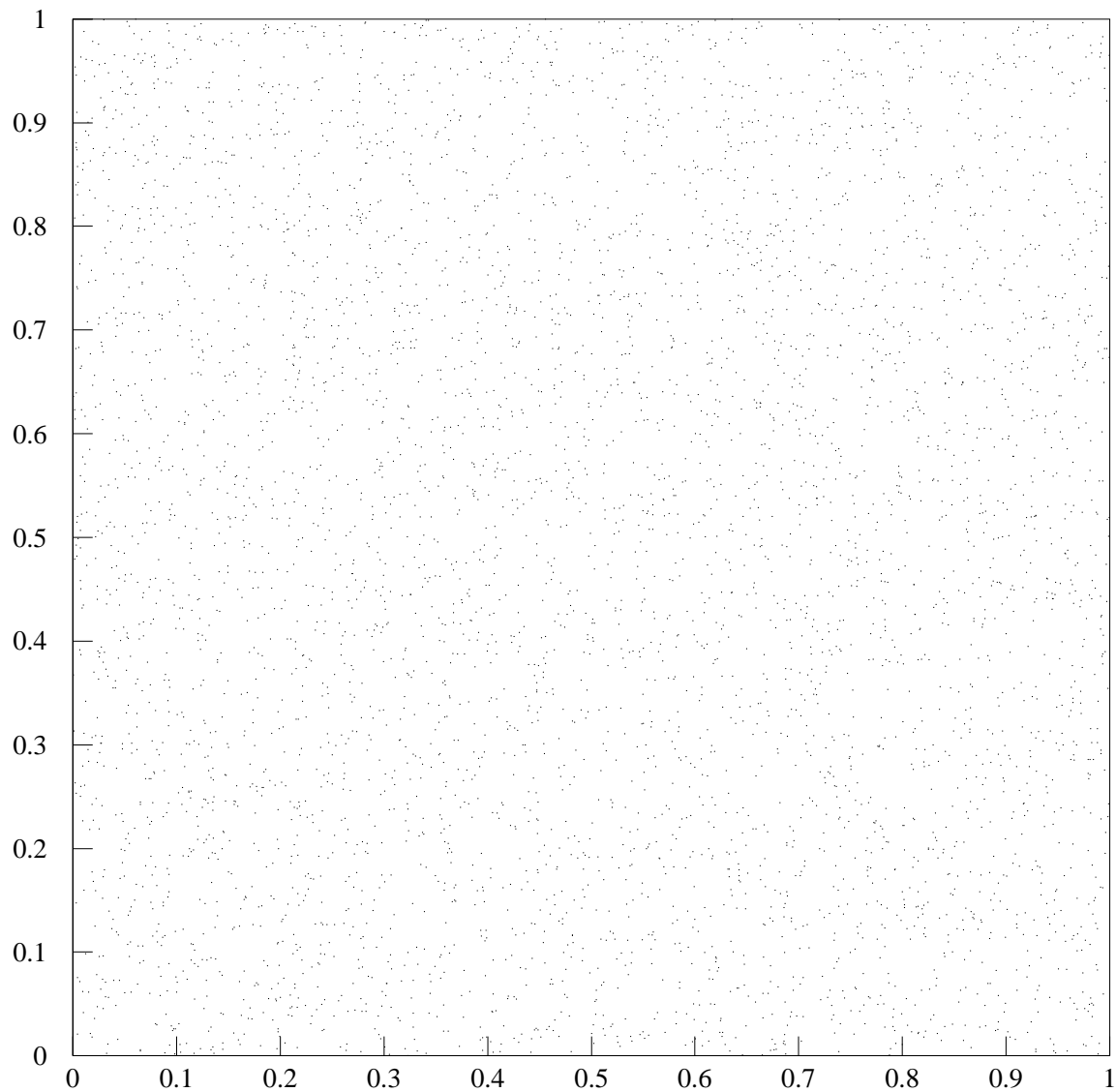


## Results from Randu: 1D distribution



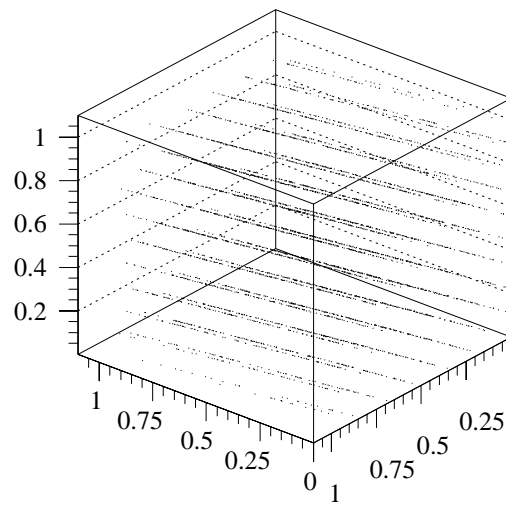
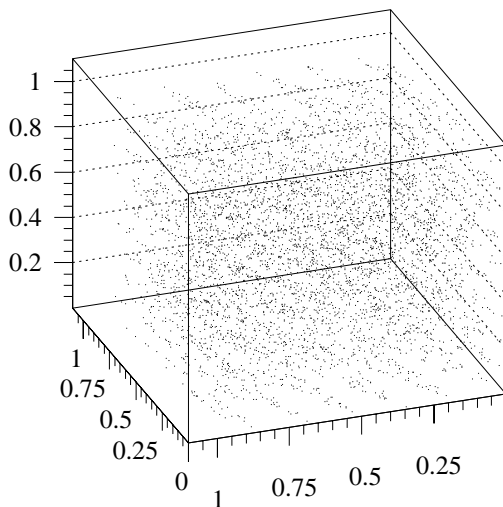
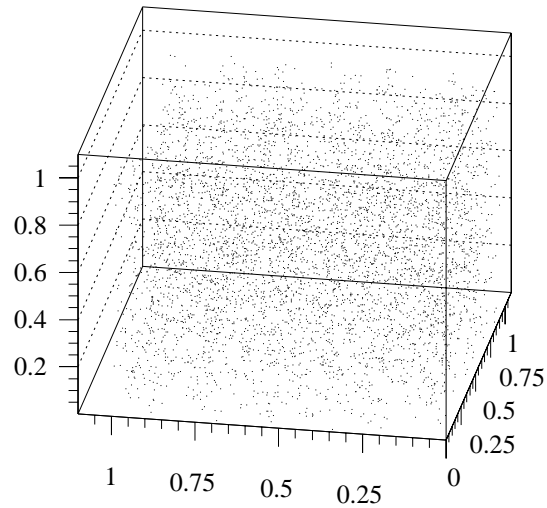
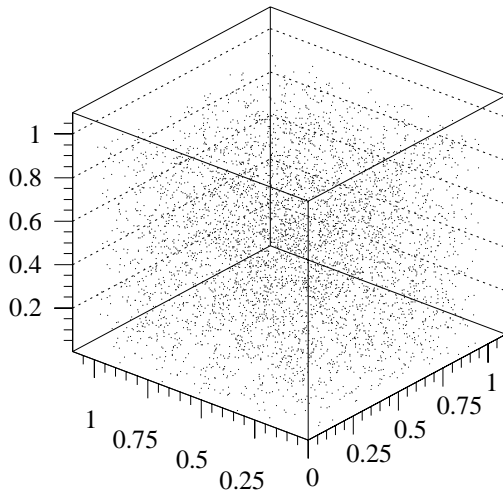
Looks okay

## Results from Randu: 2D distribution



Still looks okay

Results from Randu: 3D distribution



Problem seen when observed at the right angle!

## The Marsaglia effect

In 1968, Marsaglia published the paper,

*Random numbers fall mainly in the planes*

(Proc. Acad. Sci. 61, 25) which showed that this behaviour is present for any multiplicative congruential generator.

For a 32 bit machine, the maximum number of hyperplanes in the space of d-dimensions is:

d= 3	2953
d= 4	566
d= 6	120
d=10	41

The RANDU generator had much less than the maximum.

The replacement of the multiplier from 65539 to 69069 improves the performance significantly.



## Warning

The authors of *Numerical Recipes* have admitted that the random number generators, RAN1 and RAN2 given in the first edition, are “at best mediocre”.

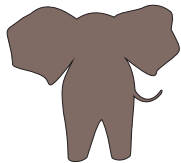
In their second edition, these are replaced by ran0, ran1, and ran2, which have much better properties.

The new routines can also be found in the recent edition of *Computers in Physics*, (Sept/Oct 1992 edition, page 522).

One way to improve the behaviour of random number generators and to increase their period is to modify the algorithm:

$$I_n = (a \times I_{n-1} + b \times I_{n-2}) \bmod m$$

Which in this case has two initial seeds and can have a period greater than  $m$ .



## RANMAR generator

This generator (available in the CERN library, KERNLIB, requires 103 initial seeds. These seeds can be set by a single integer from 1 to 900,000,000.

Each choice will generate an independent series each of period,  $\approx 10^{43}$ .

This seems to be the ultimate in random number generators!



Warning on the use of Random Number generators.

In FORTRAN, random number generators are usually used called as functions,

```
x=RAND ( IDUM)
```

Where the argument, IDUM, is not used. In fortran, a function is supposed to be a function of only the arguments, and so some compilers will try to optimise the code by removing multiple calls to random number generators.

For example

```
x=RAND ( IDUM) +RAND ( IDUM)
```

may be changed to

```
x=2 . *RAND ( IDUM)
```

This can also be a problem when the random number generator is called inside DO loops.

Solution:

Fool the optimiser by always changing the dummy argument:

```
DO 1 I=1,100
  IDUM=IDUM+1
  X=RAND (IDUM)
  . . .
1 CONTINUE
```

But don't try this if the random number generator uses the argument to save the seed for the next random number. (Numerical Recipes generators, for example)!



## Simulating Radioactive Decay

This is a truly random process: The probability of decay is constant (independent of the age of the nuclei).

The probability that a nucleus undergoes radioactive decay in time  $\Delta t$  is  $p$ :

$$p = \alpha \Delta t \quad (\text{for } \alpha \Delta t \ll 1)$$

### Problem:

Consider a system initially having  $N_0$  unstable nuclei. How does the number of parent nuclei,  $N$ , change with time?

### Algorithm:

```

LOOP from t=0 to t, step  $\Delta t$ 
  LOOP over each remaining parent nucleus
    Decide if the nucleus decays:
    IF(random # <  $\alpha \Delta t$ ) then
      reduce the number of parents by 1
    ENDIF
  END LOOP over nuclei
  PLOT or record  $N$  vs.  $t$ 
END LOOP over time
END

```

## Exercise 6

Write a program to implement the preceding algorithm. Graph the number of remaining nuclei as a function of time for the following cases:

$$N_0 = 100, \quad \alpha = 0.01 \text{ s}^{-1}, \quad \Delta t = 1 \text{ s} ;$$

$$N_0 = 5000, \quad \alpha = 0.03 \text{ s}^{-1}, \quad \Delta t = 1 \text{ s} .$$

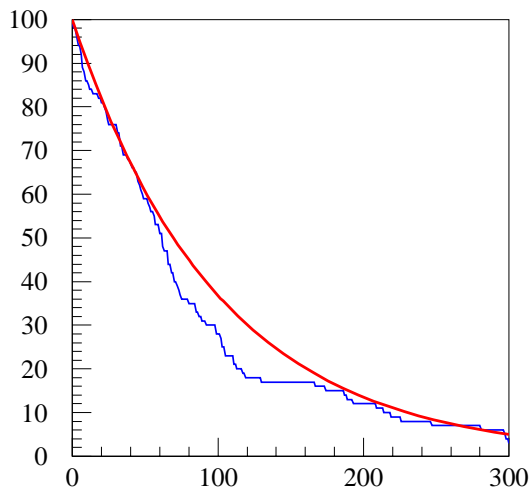
Show the results on both linear and logarithmic scales for times between 0 and 300 seconds. In addition, plot on the same graphs the expected curve, given:

$$dN = -N \alpha dt$$

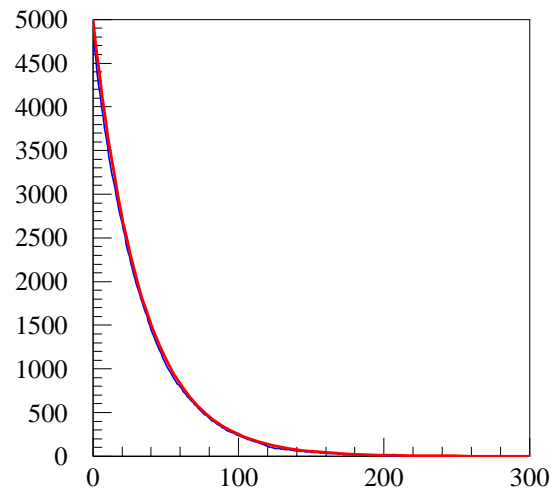
$$ie. \quad N = N_0 e^{-\alpha t}$$

## Solution to exercise 6:

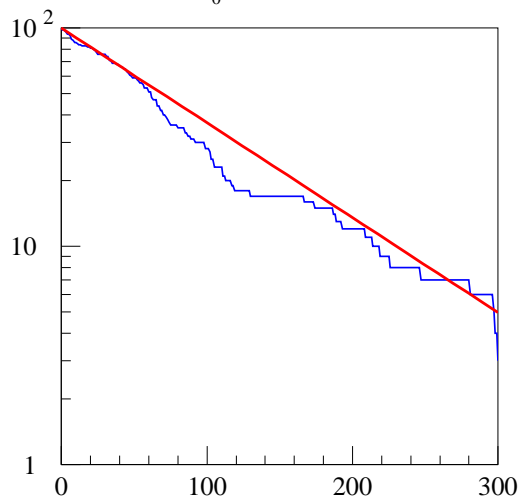
The 'experimental' results do not perfectly follow the expected curve; there are statistical fluctuations.



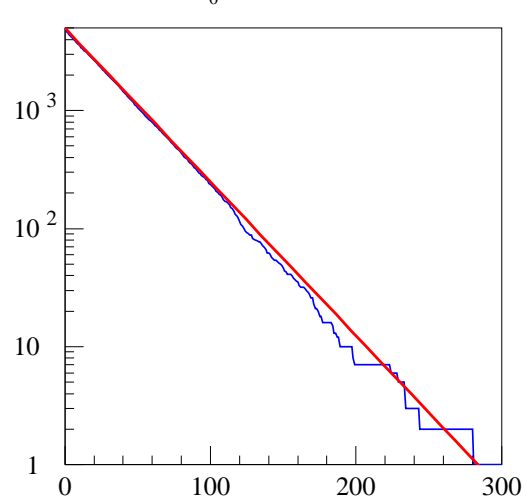
$N_0=100, \alpha=0.01$



$N_0=5000, \alpha=0.03$



$N_0=100, \alpha=0.01$



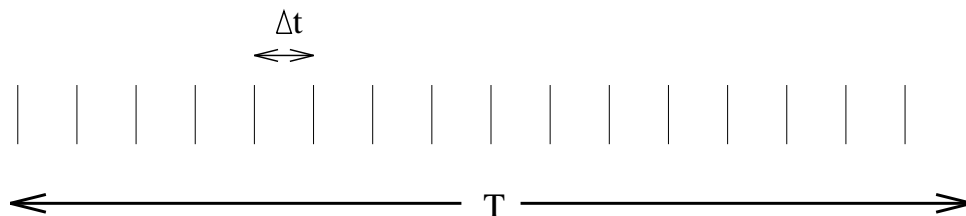
$N_0=5000, \alpha=0.03$

## Poisson Distribution

The probability of observing a total of  $n$  decays in a time interval  $T$  can be worked out as follows:

Assume the number of decays in time  $T$  is much less than the number of parent nuclei. (ie. assume constant probability to observe a decay):

Break up  $T$  into  $m$  shorter intervals, duration  $\Delta t$ :



The probability to observe 1 decay in time  $\Delta t$  is:

$$p = \beta \Delta t$$

where  $\beta = \alpha N$  as  $\Delta t$  must be small enough so that  $\beta \Delta t \ll 1$ . The probability of observing  $n$  decays in time  $T$  is therefore:

$$P = p^n (1 - p)^{m-n} \binom{m}{n} .$$

$$\begin{aligned}
 P &= p^n (1-p)^{m-n} \frac{m!}{(m-n)! n!} \\
 &= \left(\frac{\beta T}{m}\right)^n \left(1 - \frac{\beta T}{m}\right)^{m-n} \frac{m!}{(m-n)! n!}
 \end{aligned}$$

In the limit of  $\Delta t \rightarrow 0$  (ie.  $m \rightarrow \infty$ ),

$$\begin{aligned}
 \left(1 - \frac{\beta T}{m}\right)^m &\rightarrow e^{-\beta T} \\
 \left(1 - \frac{\beta T}{m}\right)^{-n} &\rightarrow 1 \\
 \frac{m!}{(m-n)!} &\rightarrow m^n
 \end{aligned}$$

The result is,

$$P = \mu^n e^{-\mu} / n!$$

where  $\mu = \beta T$ . This is known as the Poisson distribution.

## Exercise 7

Modify the program written for exercise 6 to simulate an experiment that counts the number of decays observed in a time interval,  $T$ .

Allow the experiment to be repeated and histogram the distribution of the number of decays for the following two cases:

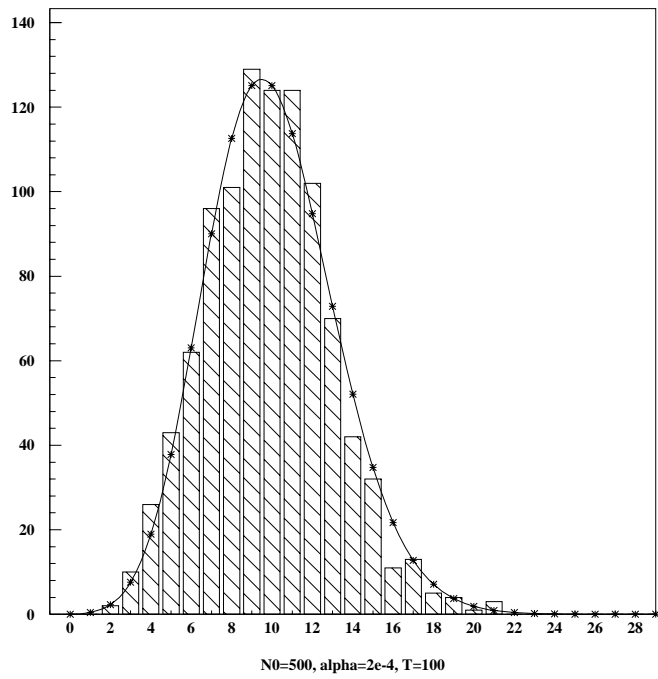
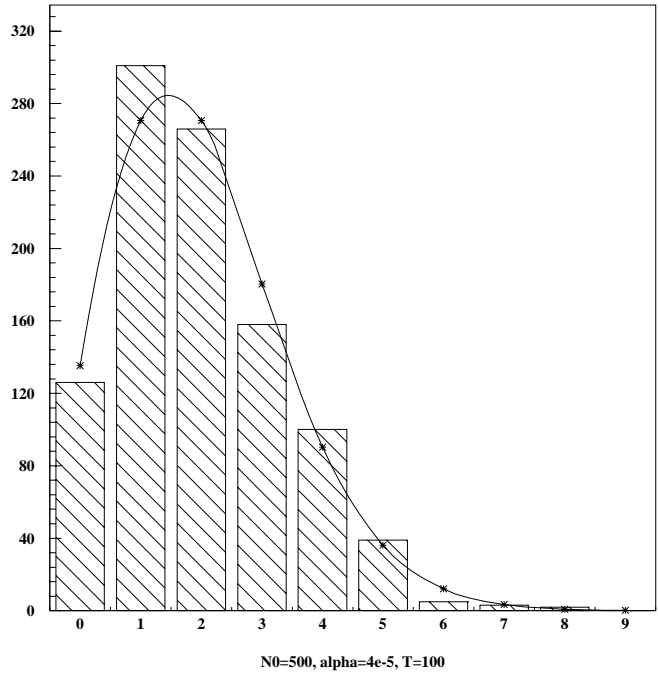
$$N_0 = 500, \alpha = 4 \times 10^{-5} \text{ s}^{-1}, \Delta t = 10 \text{ s}, T = 100 \text{ s}$$

$$N_0 = 500, \alpha = 2 \times 10^{-4} \text{ s}^{-1}, \Delta t = 10 \text{ s}, T = 100 \text{ s}$$

In each case show the distribution using 1000 experiments. Also, overlay the expected Poisson distribution.

**Question:** Are there limits on the value of  $\Delta t$  so that your program will give reliable results? Explain.

Solution to Exercise 7:



## Properties of the Poisson distribution

$$P_n = \frac{\mu^n}{n!} e^{-\mu} \quad (\mu = \alpha NT)$$

Mean value:

$$\begin{aligned} \langle n \rangle &= \sum_{n=0}^{\infty} \left( n \frac{\mu^n}{n!} e^{-\mu} \right) \\ &= \mu e^{-\mu} \sum_{n=1}^{\infty} \frac{\mu^{n-1}}{(n-1)!} \\ &= \mu e^{-\mu} \sum_{m=0}^{\infty} \frac{\mu^m}{m!} = \mu \end{aligned}$$

Variance:

$$\begin{aligned} \sigma^2 &= \sum_{n=0}^{\infty} \left( (n - \mu)^2 \frac{\mu^n}{n!} e^{-\mu} \right) \\ &= \sum_{n=0}^{\infty} \left( (n^2 - 2n\mu + \mu^2) \frac{\mu^n}{n!} e^{-\mu} \right) \end{aligned}$$



Do each term individually,

$$\begin{aligned} \sum_{n=0}^{\infty} \left( n^2 \frac{\mu^n}{n!} e^{-\mu} \right) &= \sum_{n=1}^{\infty} \left( n \frac{\mu^{n-1}}{(n-1)!} e^{-\mu} \right) \mu \\ &= \sum_{n=0}^{\infty} \left( (n+1) \frac{\mu^n}{n!} e^{-\mu} \right) \mu \\ &= (\mu + 1)\mu \end{aligned}$$

$$\sum_{n=0}^{\infty} \left( -2n\mu \frac{\mu^n}{n!} e^{-\mu} \right) = -2\mu^2$$

$$\sum_{n=0}^{\infty} \left( \mu^2 \frac{\mu^n}{n!} e^{-\mu} \right) = \mu^2$$

So,  $\sigma^2 = \mu^2 + \mu - 2\mu^2 + \mu^2 = \mu$ .

Hence if  $n$  decays are observed, the 1 standard deviation uncertainty is  $\sqrt{n}$ . (This is also true for any other variable that follows the Poisson distribution.)

Many observables follow the Poisson distribution: Anything whose probability of occurring is constant in time.

For example:

- number of observed events when efficiency is constant
- number of entries in a histogram bin

Some measurements lead to non-Poisson distributions:

For example:

- number of radioactive decays observed in a fixed time interval, when there is a significant reduction of parent nuclei
- number of radioactive decays observed in a fixed time interval, when there is significant deadtime. (ie. the detector is not active for some period after an event is recorded)

## Gaussian (or Normal) Distribution

This is the most important distribution in statistical analysis.

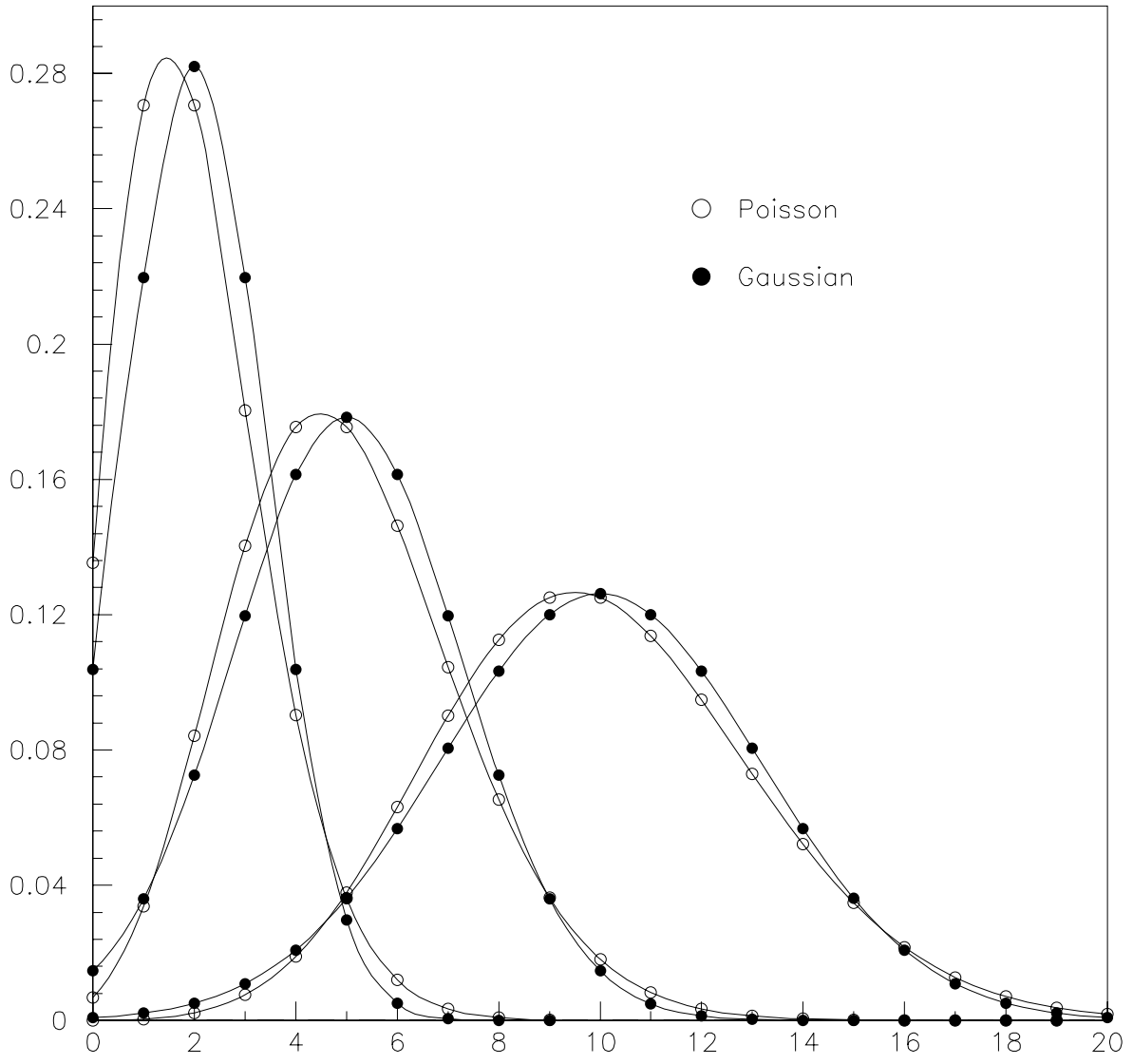
$$G(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The mean of the distribution is  $\mu$  and the variance is  $\sigma^2$ .

For large  $\mu$ , the Poisson distribution approaches the Gaussian distribution (with  $\sigma^2 = \mu$ ).

The Gaussian distribution is a reasonable approximation of the Poisson distribution even for  $\mu$  as small as 5.

Comparison of Poisson and Gaussian distributions:



## Binomial Distribution

The binomial distribution describes the results of repeated experiments which has only two possible outcomes.

Suppose a radioactive source is monitored for a time interval  $T$ . There is a probability  $p$  that one or more disintegrations would be detected in that time interval. If a total of  $m$  intervals were recorded, the probability that  $n$  of them had at least one decay is

$$P = p^n (1 - p)^{m-n} \binom{m}{n} .$$

The mean of this distribution is:  $np$

The variance of this distribution is:  $np(1 - p)$

## Simulating General Distributions

The simple simulations considered so far, only required a random number sequence that is uniformly distributed between 0 and 1. More complicated problems generally require random numbers generated according to specific distributions.

For example, the radioactive decay of a large number of nuclei (say  $10^{23}$ ), each with a tiny decay probability, cannot be simulated using the methods developed so far. It would be far too inefficient and require very high numerical precision.

Instead, a random number generated according to a Poisson distribution could be used to specify the number of nuclei that disintegrate in some time  $T$ .

Random numbers following some special distributions, like the Poisson distribution, can be generated using special purpose algorithms, and efficient routines can be found in various numerical libraries.

If a special purpose generator routine is not available, then use a general purpose method for generating random numbers according to an arbitrary distribution.

## Rejection Technique

**Problem:** Generate a series of random numbers,  $x_i$ , which follow a distribution function  $f(x)$ .

In the rejection technique, a trial value,  $x_{\text{trial}}$  is chosen at random. It is accepted with a probability proportional to  $f(x_{\text{trial}})$ .

**Algorithm:**

Choose trial  $x$ , given a uniform random number  $\lambda_1$ :

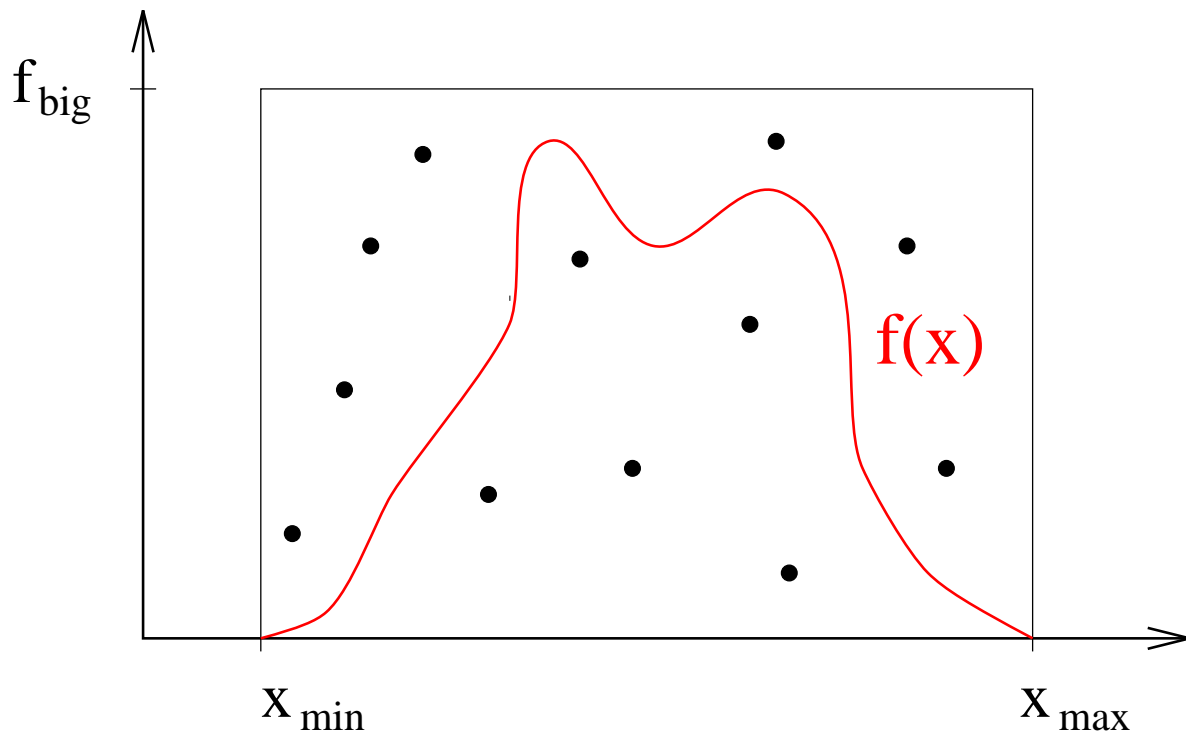
$$x_{\text{trial}} = x_{\text{min}} + (x_{\text{max}} - x_{\text{min}})\lambda_1$$

Decide whether to accept the trial value:

$$\text{if } f(x_{\text{trial}}) > \lambda_2 f_{\text{big}} \text{ then accept}$$

where  $f_{\text{big}} \geq f(x)$  for all  $x$ ,  $x_{\text{min}} \leq x \leq x_{\text{max}}$ . Repeat the algorithm until a trial value is accepted.

This algorithm can be visualized as throwing darts:



This procedure also gives an estimate of the integral of  $f(x)$ :

$$I = \int_{x_{\text{min}}}^{x_{\text{max}}} f(x) dx \approx \frac{n_{\text{accept}}}{n_{\text{trial}}} f_{\text{big}} (x_{\text{max}} - x_{\text{min}})$$



The 1 standard deviation uncertainty can be derived using the variance of the binomial distribution:

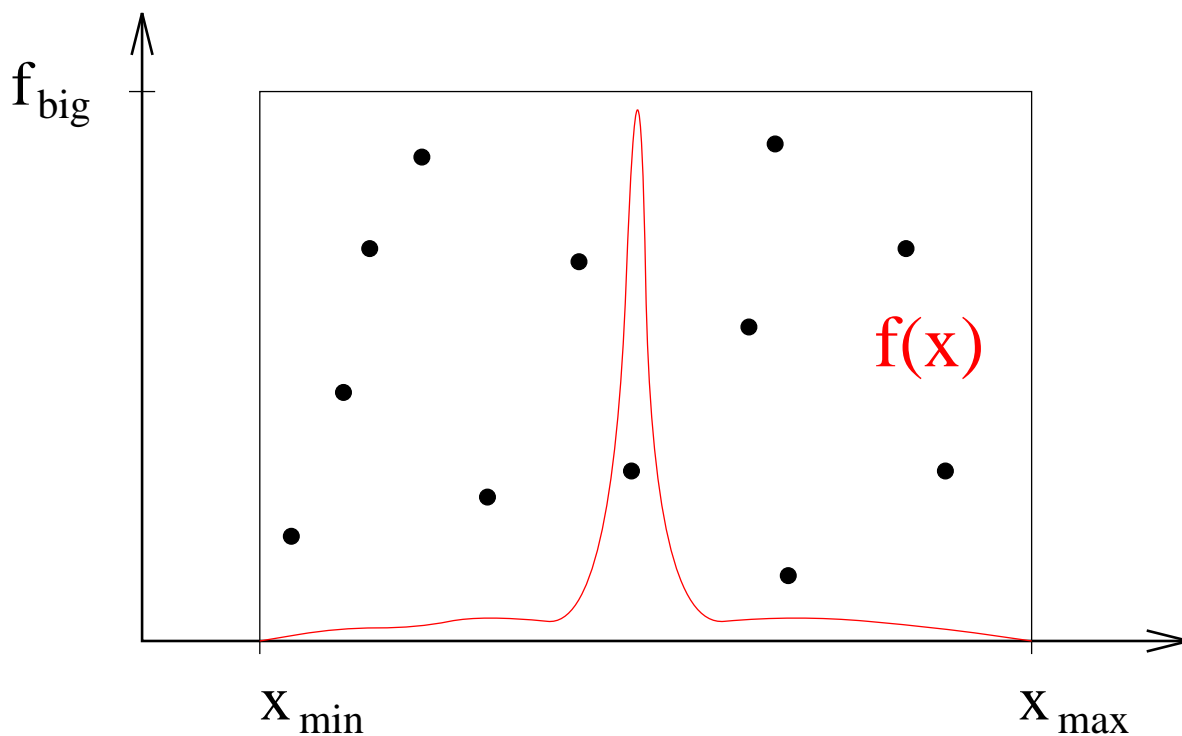
$$\delta N_{\text{accept}} = \sqrt{p(1-p)N_{\text{trial}}} \quad p = \frac{N_{\text{accept}}}{N_{\text{trial}}}$$

$$\begin{aligned} \left(\frac{\delta I}{I}\right)^2 &= \left(\frac{\delta N_{\text{accept}}}{N_{\text{accept}}}\right)^2 \\ &= \frac{N_{\text{accept}}}{N_{\text{trial}}} \left(1 - \frac{N_{\text{accept}}}{N_{\text{trial}}}\right) N_{\text{trial}} \frac{1}{N_{\text{accept}}^2} \\ &= \frac{1}{N_{\text{accept}}} - \frac{1}{N_{\text{trial}}} \\ &= \frac{1}{N_{\text{trial}}} \left(\frac{1-p}{p}\right) \end{aligned}$$

So the relative accuracy only improves with  $N_{\text{trial}}^{-\frac{1}{2}}$

The rejection algorithm is not efficient if the distribution has one or more large peaks (or poles).

In this case trial events are seldomly accepted:



In extreme cases, where there is a pole,  $f_{\text{big}}$  cannot be specified. This algorithm doesn't work when the range of  $x$  is  $(-\infty, +\infty)$ . A better algorithm is needed...

## Inversion Technique

This method is only applicable for relatively simple distribution functions:

- First normalize the distribution function, so that it becomes a probability distribution function (PDF).
- Integrate the PDF analytically from the minimum  $x$  to an arbitrary  $x$ . This represents the probability of choosing a value less than  $x$ .
- Equate this to a uniform random number, and solve for  $x$ . The resulting  $x$  will be distributed according to the PDF.

In other words, solve the following equation for  $x$ , given a uniform random number,  $\lambda$ :

$$\frac{\int_{x_{\min}}^x f(x) dx}{\int_{x_{\min}}^{x_{\max}} f(x) dx} = \lambda$$

This method is fully efficient, since each random number  $\lambda$  gives an  $x$  value.

### Examples of the inversion technique

1) generate  $x$  between 0 and 4 according to  $f(x) = x^{-\frac{1}{2}}$ :

$$\frac{\int_0^x x^{-\frac{1}{2}} dx}{\int_0^4 x^{-\frac{1}{2}} dx} = \lambda$$

$$\frac{1}{2}x^{\frac{1}{2}} = \lambda$$

$\Rightarrow$  generate  $x$  according to  $x = 4\lambda^2$

2) generate  $x$  between 0 and  $\infty$  according to  $f(x) = e^{-x}$ :

$$\frac{\int_0^x e^{-x} dx}{\int_0^{\infty} e^{-x} dx} = \lambda$$

$$1 - e^{-x} = \lambda$$

$\Rightarrow$  generate  $x$  according to  $x = -\ln(1 - \lambda)$

Note that the simple rejection technique would not work for either of these examples.

## Exercise 8

Write a program that generates the value  $\theta$  according to the distribution function:

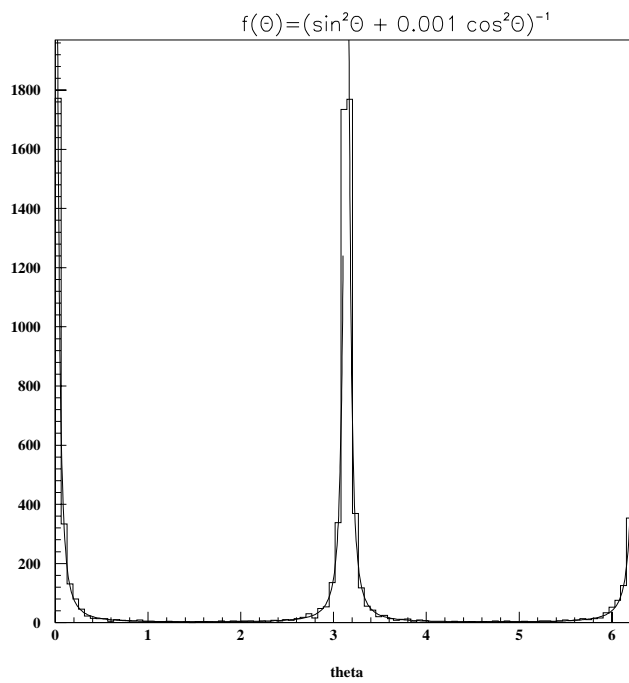
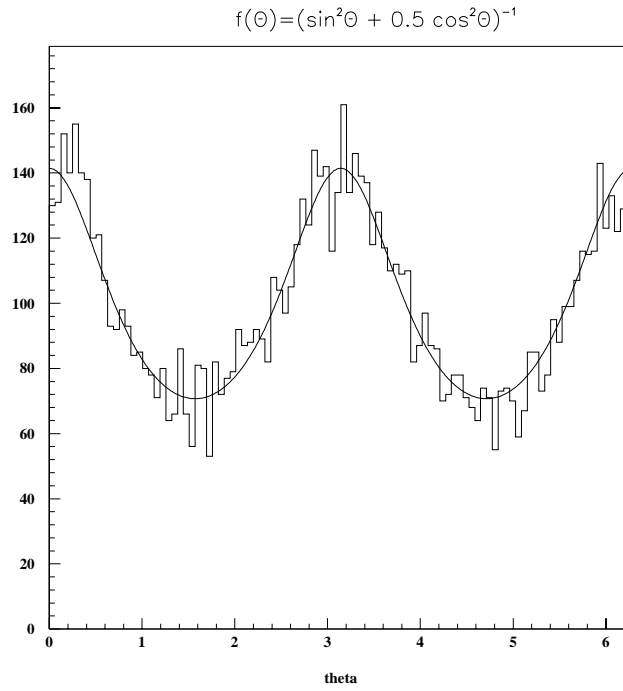
$$f(\theta) = (\sin^2 \theta + a \cos^2 \theta)^{-1}$$

in the range  $0 \leq \theta \leq 2\pi$ .

Compare the rejection technique and the inversion technique:

- Generate 10000 values for each method using  $a = 0.5$  and also for  $a = 0.001$ .
- Plot the results for each (4 plots) and overlay the distribution curve,  $f(\theta)$ , properly normalized.
- Compare the CPU time required for the 4 runs.

Solution to Exercise 8:



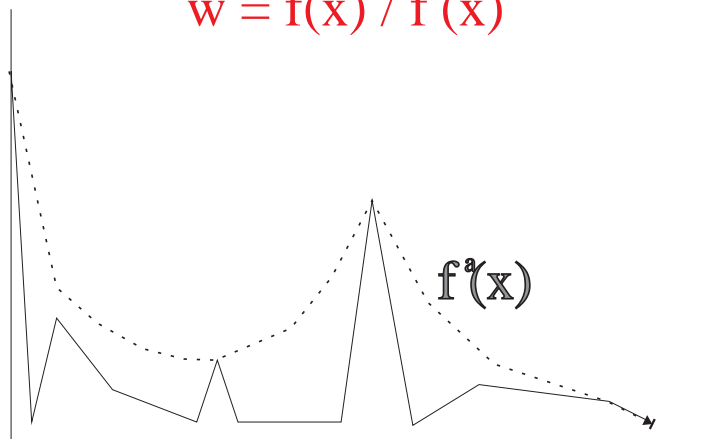
What if the rejection technique is impractical and you can't invert the integral of the distribution function?

### Importance Sampling

Replace the distribution function,  $f(x)$ , by an approximate form,  $f^a(x)$ , for which the inversion technique can be applied.

Generate trial values for  $x$  with the inversion technique according to  $f^a(x)$ , and accept the trial value with the probability proportional to the weight:

$$w = f(x) / f^a(x)$$



The rejection technique is just the special case where  $f^a(x)$  is chosen to be a constant.

Example:

Generate  $x$  according to  $f(x) = (1+x)x^{-1/2}$   
for the range  $0 < x < 1$ .

There is clearly a problem at  $x$  near 0.

$f'(x)$  needs to be chosen so the weights for the  
trial values of  $x$  are well behaved,

$$w = f(x)/f'(x)$$

Try  $f'(x) = x^{-1/2}$ , then  $w=1+x$

Procedure:

Generate trial  $x$ :  $x = \lambda_1^2$

Decide to accept: if  $(1+x) > \lambda_2 w_{\max}$  accept

In this case,  $w_{\max}=2$ , but in more complicated cases,  
you may need to run the program to find the  
maximum weight generated, and then pick a value a  
little larger, and rerun.



Note: the integral can be evaluated as before

$$I = \int_{x_{\min}}^{x_{\max}} f(x) dx = \frac{n_{\text{accept}}}{n_{\text{trial}}} w_{\max} I_a$$

where  $I_a = \int_{x_{\min}}^{x_{\max}} f^a(x) dx$ .

But the integral can be found more efficiently (ie. more accurately for the same amount of CPU), by using the weights of all trial values:

$$\begin{aligned} I &= \int_{x_{\min}}^{x_{\max}} f(x) dx = \int_{x_{\min}}^{x_{\max}} \frac{f(x)}{f^a(x)} f_a(x) dx \\ &= \int_{x_{\min}}^{x_{\max}} w(x) f^a(x) dx \end{aligned}$$

But,  $\int_{x_{\min}}^x f^a(x) dx / I_a = \lambda$ , so  $f^a(x) dx = I_a d\lambda$

$$I = \int_0^1 w(\lambda) I_a d\lambda = I_a \frac{1}{n_{\text{trial}}} \sum_i w = I_a \langle w \rangle$$

And the one standard deviation uncertainty is,

$$\left( \frac{\delta I}{I} \right)^2 = \frac{1}{n_{\text{trial}}} \frac{\langle w^2 \rangle - \langle w \rangle^2}{\langle w \rangle^2}$$

## Generating random numbers according to the Gaussian distribution.

---

There are 2 ways to handle this special case.

### 1) Central limit theorem

---

“The sum of a large number of random numbers will approach a Gaussian distribution”

For a uniform distribution from 0 to 1,  
the mean value is 1/2

and the variance is

$$\sigma^2 = \int (x-1/2)^2 dx = 1/12$$

So just add 12 random numbers and subtract 6.  
The mean will be 0 and the variance will be 1.

This algorithm is coded in RG32 in the  
CERN library.

## 2) 2 D gaussian

Consider the 2 dimensional Gaussian dist:

$$\begin{aligned} f(x,y) dx dy &= e^{-x^2/2} dx e^{-y^2/2} dy \\ &= e^{-(x^2+y^2)/2} dx dy \end{aligned}$$

Let  $r^2 = x^2 + y^2$  and  $\theta = \tan^{-1} y/x$

then,  $dx dy = r dr d\theta$

$$f(r,\theta) dr d\theta = e^{-r^2/2} r dr d\theta$$

Let  $u = r^2/2$  then  $du = r dr$

$$f(u,\theta) du d\theta = e^{-u} du d\theta$$

So generate  $u$  between 0 and  $\infty$  according to  $e^{-u}$  and  $\theta$  between 0 and  $2\pi$  (uniformly):

$$u = -\log(1-\lambda_1)$$

$$r = (2u)^{1/2}$$

$$\theta = 2\pi\lambda_2$$

$$x = r \cos(\theta)$$

$$y = r \sin(\theta)$$

This is coded in the CERN library function, RANNOR, but a method 5 times faster is available in NORRAN.

## Multidimensional Simulation

Simulating a distribution in more than 1 dimension:

---

If the distribution is separable, the variables are uncorrelated, and hence each can be generated as before:

For example, if  $f(x,y) = g(x) h(y)$

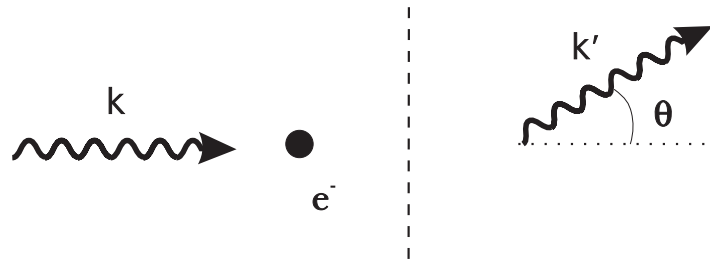
then generate  $x$  according to  $g(x)$  and  $y$  according to  $h(y)$ .

Otherwise, the distributions along each dimension needs to be calculated:

$$D_x(x) = \int_{y_{\min}}^{y_{\max}} f(x,y) dy$$

Typically, you will need to choose an approximation of the distribution,  $f^a(x,y)$  so the integrals,  $\int f^a(x,y)dx$  and  $\int f^a(x,y)dy$  are invertable. The weights for trial events are given by,  $w = f(x,y) / f^a(x,y)$  and the integral can be evaluated as before, using the weights of all trial events. (Event =  $x$  and  $y$  pair)

## Simulation of Compton Scattering



The energy of the final state photon is given by

$$k' = \frac{k}{1 + (k/m)(1 - \cos \theta)}$$

The differential cross section is:

$$\frac{d\sigma}{d\Omega} = \frac{\alpha^2}{2m^2} \left(\frac{k'}{k}\right)^2 \left(\frac{k'}{k} + \frac{k}{k'} - \sin^2 \theta\right)$$

O. Klein, Y. Nishina, Z. Physik, 52, 853 (1929)

The angular distribution of the photon is:

$$\sigma(\theta, \phi) d\theta d\phi = \frac{\alpha^2}{2m^2} \times \left( \left(\frac{k'}{k}\right)^3 + \left(\frac{k'}{k}\right) - \left(\frac{k'}{k}\right)^2 \sin^2 \theta \right) \sin \theta d\theta d\phi$$

The azimuthal angle,  $\phi$ , can be generated independantly from  $\theta$ , by simply:  $\phi = 2\pi \lambda_1$ .

To generate the polar angle,  $\theta$ , an approximation is needed. Note that for  $k \gg m$ , the cross section is sharply peaked at small angles. Also, note that  $k' < k$ , so the second term is the dominant term in the cross section formula.

A good approximation to the cross section is,

$$\begin{aligned}\sigma^a(\theta, \phi) d\theta d\phi &= \frac{\alpha^2}{2m^2} \left(\frac{k'}{k}\right) \sin \theta d\theta d\phi \\ &= \frac{\alpha^2}{2m^2} \left(1 + \frac{k}{m} u\right)^{-1} du d\phi\end{aligned}$$

where  $u = (1 - \cos \theta)$ .

$u$  is generated according to:

$$u = \frac{m}{k} \left[ \left(1 + 2\frac{k}{m}\right)^{\lambda_2} - 1 \right]$$

Be careful when  $k \ll m$ ; this procedure would not generate  $u$  properly, due to roundoff errors. Similarly, it is much better to generate  $u = (1 - \cos \theta)$  than  $\cos \theta$ , when there is a pole at  $\theta = 0$ .



## Exercise 9

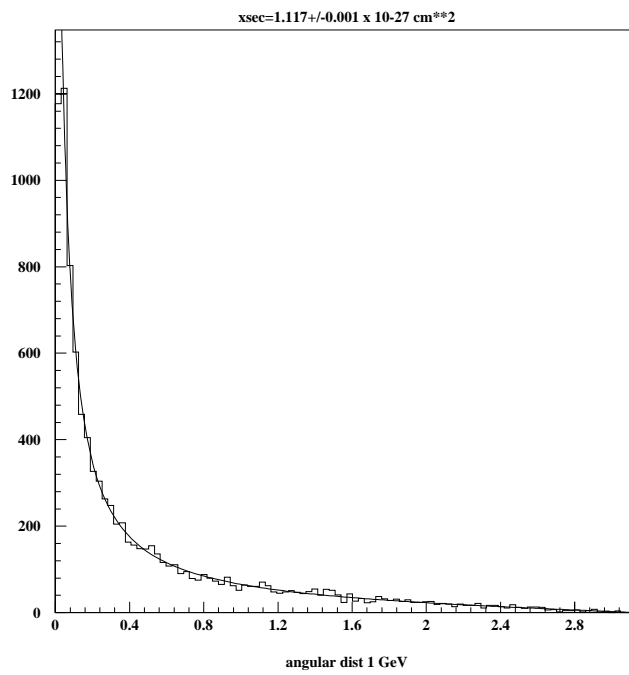
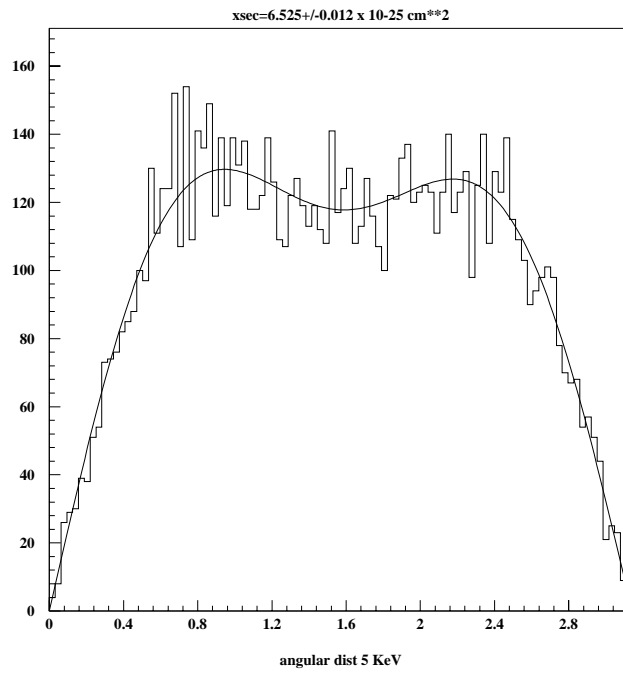
Write a Monte Carlo program that generates Compton scattering events.

The program should ask for the number of events to generate and the photon energy. Show the distribution of the scattering angle of the photon (compared to the Klein Nishina formula) and give the total cross section (ie. use the same program to evaluate the integral and its uncertainty) for the following four cases:

$k=5$  keV,  $k=2$  MeV,  $k=1$  GeV,  $k=1$  TeV

in each case generate 10000 events.

Solution to Exercise 9:





## Photon transport in matter

With this program, and others that simulate the photoelectric effect, pair production, etc., you could produce a program that simulates the interaction of photons with matter:

### Algorithm:

Break path into small steps:

For each step decide if an interaction takes place (given the total cross section for each possible interaction).

Simulate the interaction, ie. give photon new momentum vector or possibly produce an  $e^+e^-$  pair, which then would be followed, etc.

Such programs already exist. For example:

EGS (SLAC)  
GEANT (CERN)

You may use these to simulate photon transport in a particular sample you are testing or to simulate the response of your detector.

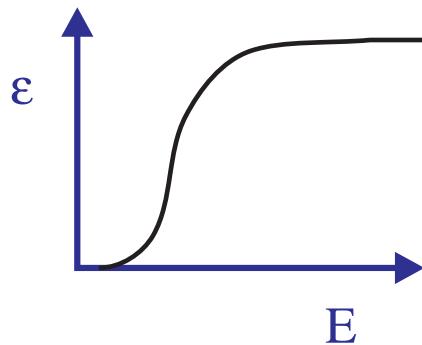
### Detector response

It is often sufficient to simulate the general properties of your detector: efficiency, resolution, bias, offset.

### Efficiency

From measurements from well understood sources, the efficiency as a function of energy (and maybe position) can be found.

For example:

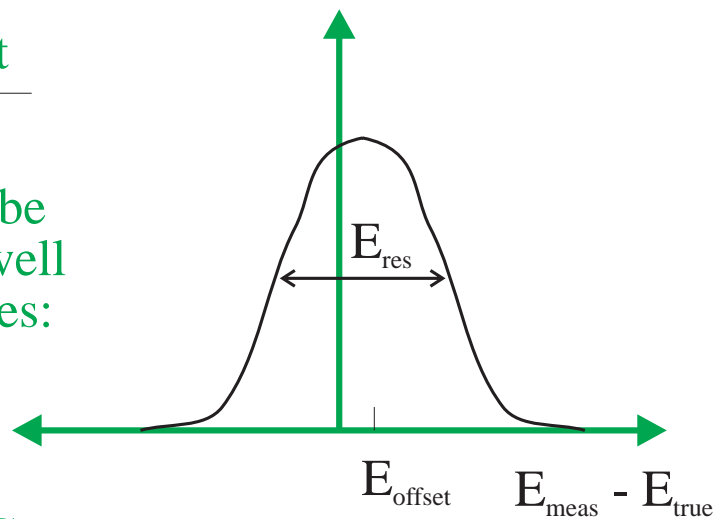


Once  $\epsilon$  is known, select events that are missed with a probability of  $\epsilon$ :

If  $\lambda > \epsilon$  then event is not observed.

Resolution, offset

Again, these can be measured using well understood sources:



$$E_{\text{meas}} = E_{\text{true}} + E_{\text{res}} G_{\lambda} + E_{\text{offset}}$$

↑ Gaussian random number

Background, noise

Simulate the observed energy distribution when no source is present.