993SM - Laboratory of Computational Physics lecture 3 March 16, 2022

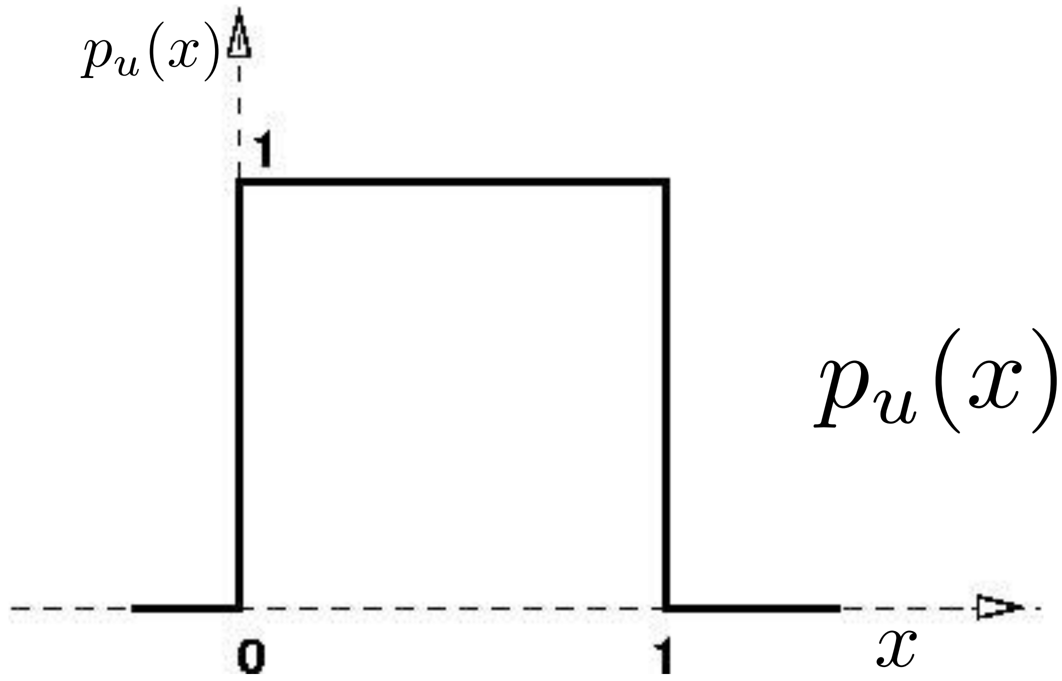
Maria Peressi
Università degli Studi di Trieste - Dipartimento di Fisica
Sede di Miramare (Strada Costiera 11, Trieste)
e-mail: peressi@units.it
tel.: +39 040 2240242

I) Random numbers with non uniform distributions and II) random processes

M. Peressi - UniTS - Laurea Magistrale in Physics
Laboratory of Computational Physics - Unit III

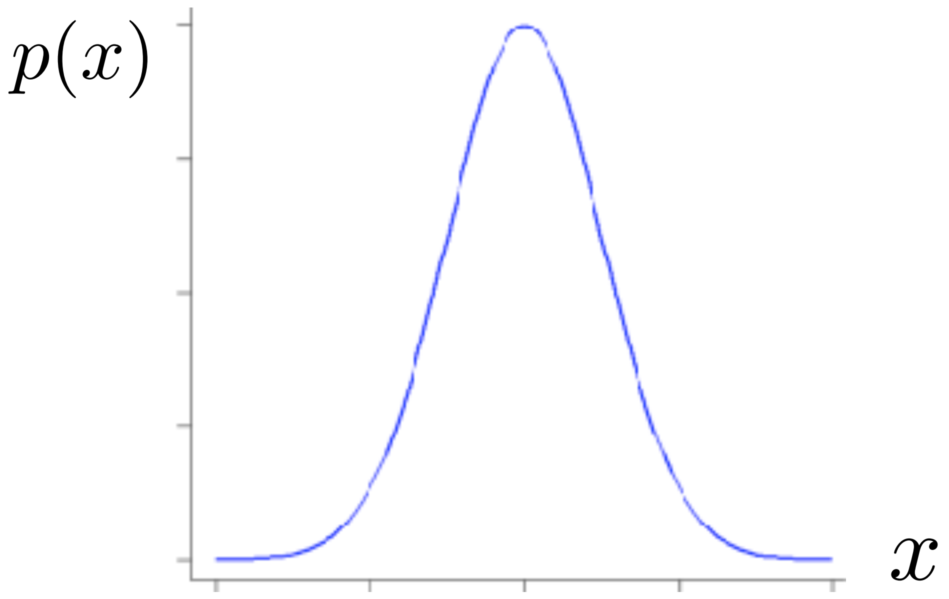
last lecture:

generation of real (pseudo)random numbers with uniform distribution in [0;1[



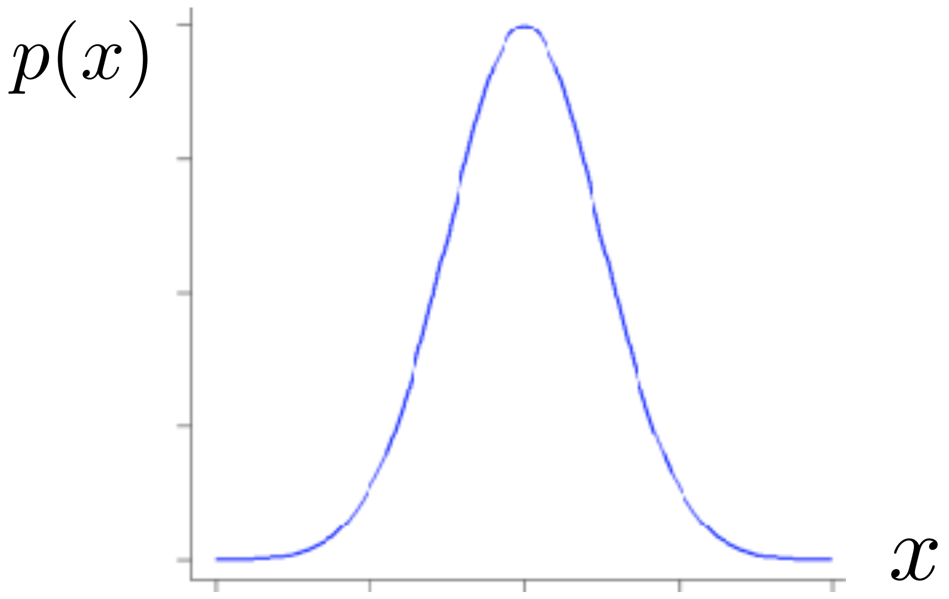
$$p_u(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases}$$

Part I - Random numbers with non uniform distributions:



How can we generate random numbers with a given distribution p(x) ?

Part I - Random numbers with non uniform distributions:



1) inverse transformation method (general)
2) rejection method (even more general)
3) some "ad hoc" methods: the Box-Muller algorithm for the gaussian distribution

Non uniform random numbers distribution: 1) inverse transformation method (general)

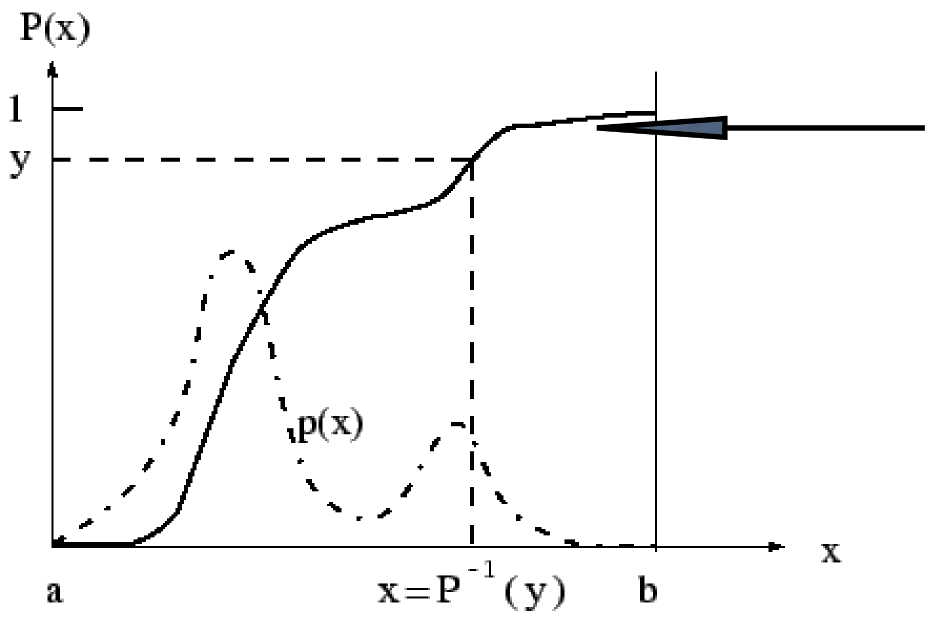
Problem: Generate sample of a random variable (or *variate*) x with a given distribution p .

Solution: 2-step process

- Generate a random variate uniformly distributed in $[0, 1]$ .. also called a *random number*

- Use an appropriate transformation to convert the random number to a random variate of the correct distribution

Non uniform random numbers distribution: 1) inverse transformation method - algorithm



$$\int_{-\infty}^{+\infty} p(x)dx = 1$$

Let $p(x)$ be a desired distribution, and $y = P(x) = \int_{-\infty}^{x} p(x')dx'$ the corresponding *cumulative distribution.* Assume that $P^{-1}(y)$ is known.

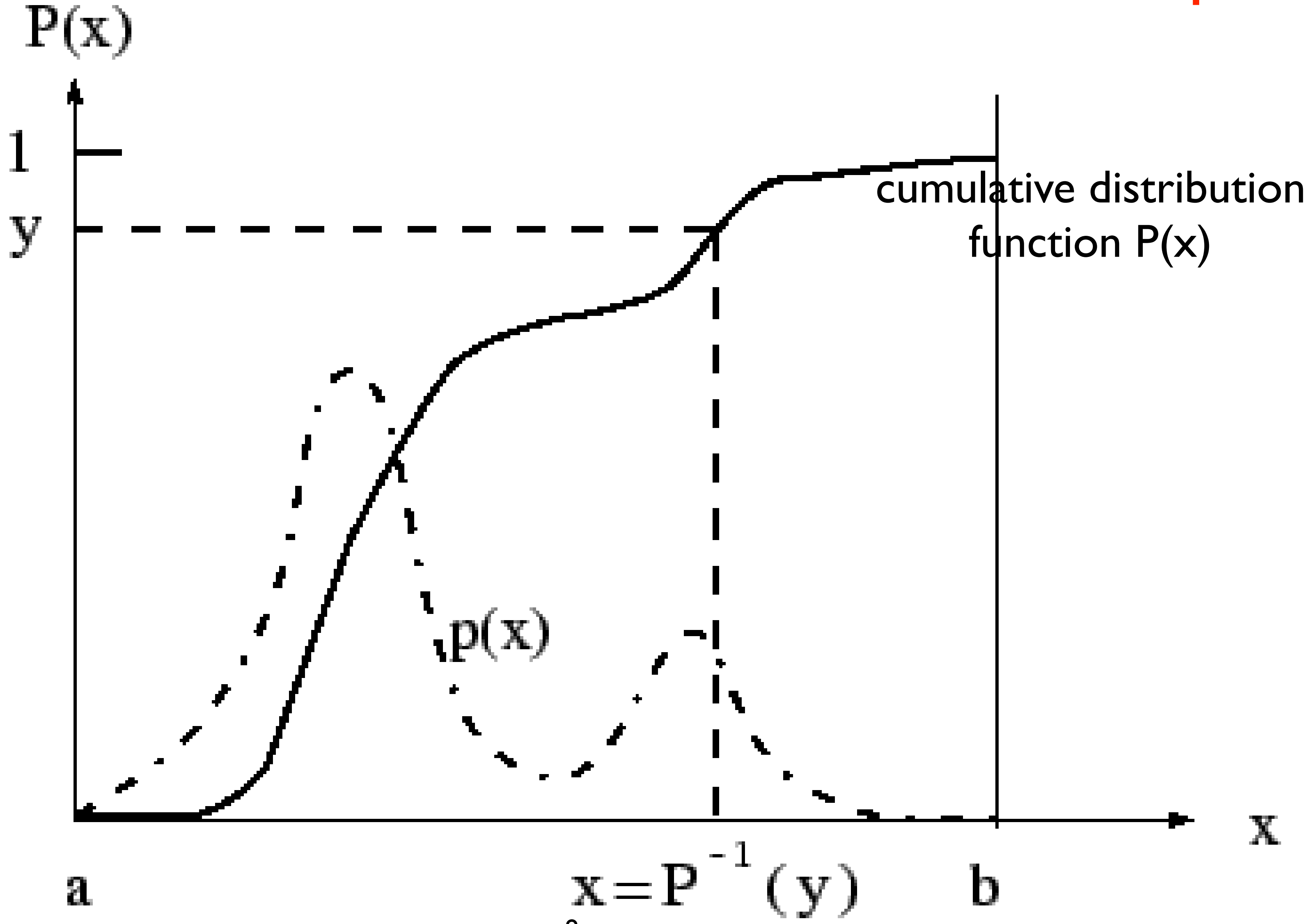
- Sample y from an equidistribution in the interval (0,1). (i.e., use $p_u(y)$)
- Compute $x = P^{-1}(y)$.

The variable x then has the desired probability density $p(x)$.

$$y = P(x) \Longrightarrow dy = dP(x) \Longrightarrow p_u(y)dy = dP(x) \ \ (\text{since } p_u(y) = 1 \text{ for } 0 \leq y \leq 1)$$

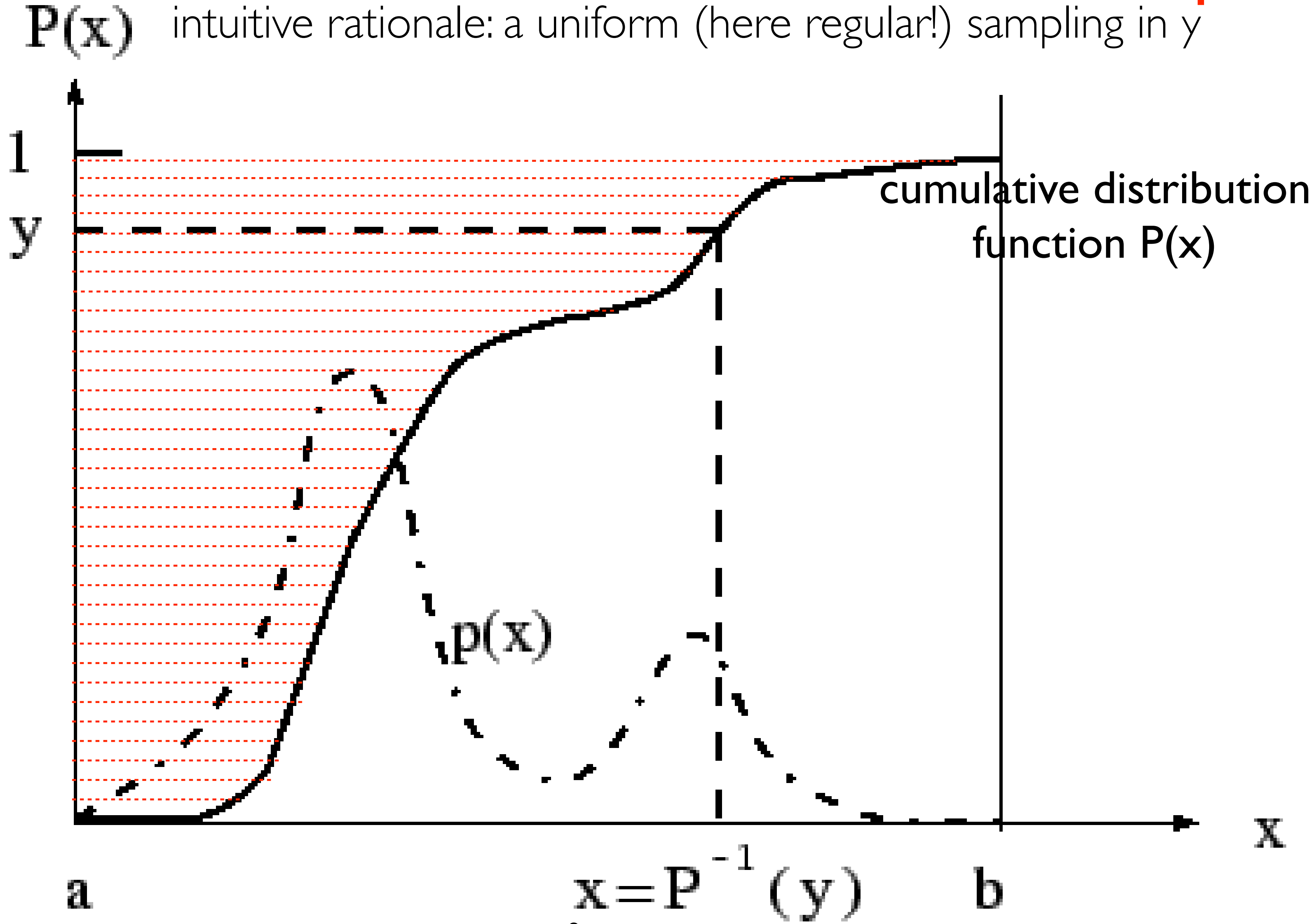
But : $\quad dP(x) = p(x)dx, \quad$ therefore $\ p(x)dx = p_u(y)dy$

Non uniform random numbers distribution:
1) inverse transformation method - the concept



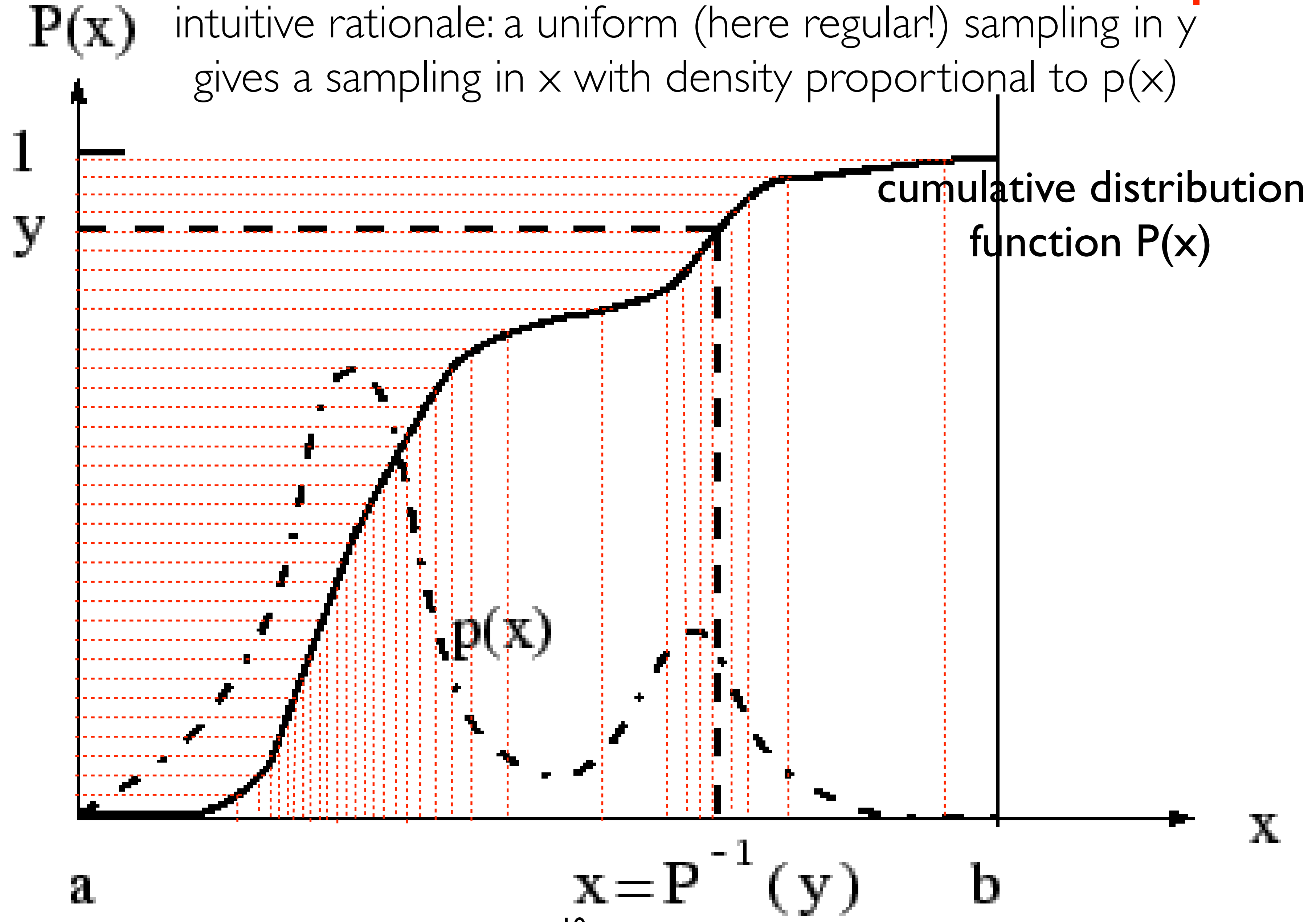
Non uniform random numbers distribution: 1) inverse transformation method - the concept

intuitive rationale: a uniform (here regular!) sampling in y



Non uniform random numbers distribution: 1) inverse transformation method - the concept

intuitive rationale: a uniform (here regular!) sampling in y gives a sampling in x with density proportional to p(x)

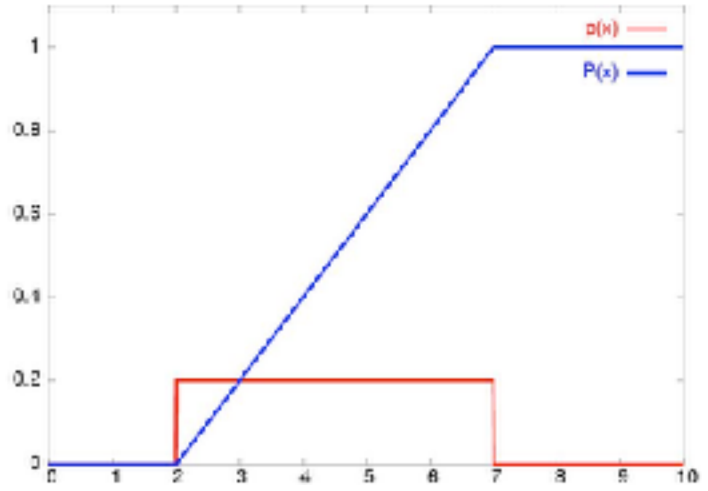


Non uniform random numbers distribution: 1) inverse transformation method - examples

1) $$p(x) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & \text{otherwise} \end{cases}$$

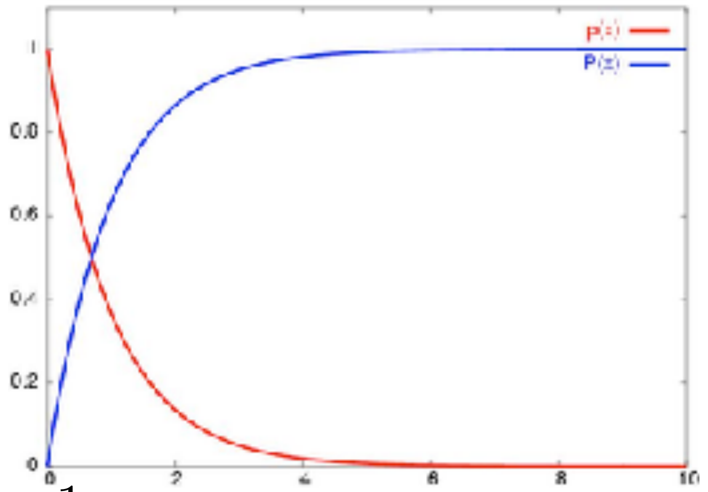
$$y = \quad P(x) = \begin{cases} 0 & x \le a \\ \int_a^x \frac{1}{b-a} dx' = \frac{x-a}{b-a} & a \le x \le b \\ 1 & x > b \end{cases}$$

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



2) $$p(x) = \begin{cases} 0 & x \le 0 \\ ae^{-ax} & x \ge 0 \end{cases}$$

$$y = \quad P(x) = \begin{cases} 0 & x \le 0 \\ 1 - e^{-ax} & x \ge 0 \end{cases}$$



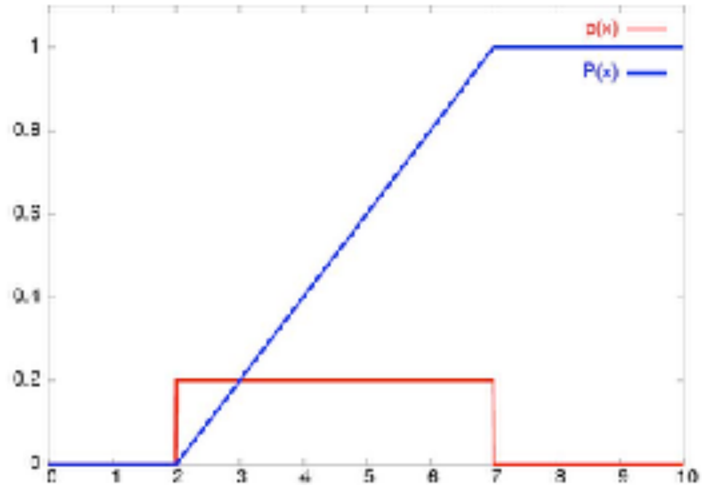
$$x = -\frac{1}{a}\ln(1-y) \quad \text{or (same distribution!)} \quad x = -\frac{1}{a}\ln y$$

Non uniform random numbers distribution: 1) inverse transformation method - examples

1) $$p(x) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & \text{otherwise} \end{cases}$$

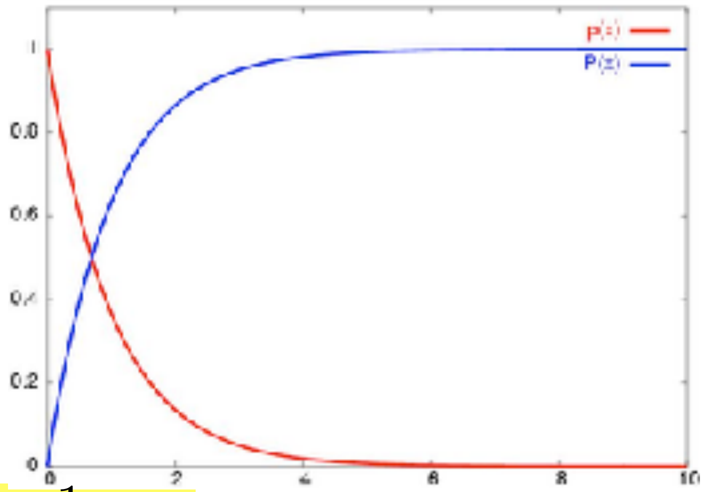
$$y = P(x) = \begin{cases} 0 & x \le a \\ \int_a^x \frac{1}{b-a} dx' = \frac{x-a}{b-a} & a \le x \le b \\ 1 & x > b \end{cases}$$

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



2) $$p(x) = \begin{cases} 0 & x \le 0 \\ ae^{-ax} & x \ge 0 \end{cases}$$

$$y = P(x) = \begin{cases} 0 & x \le 0 \\ 1 - e^{-ax} & x \ge 0 \end{cases}$$

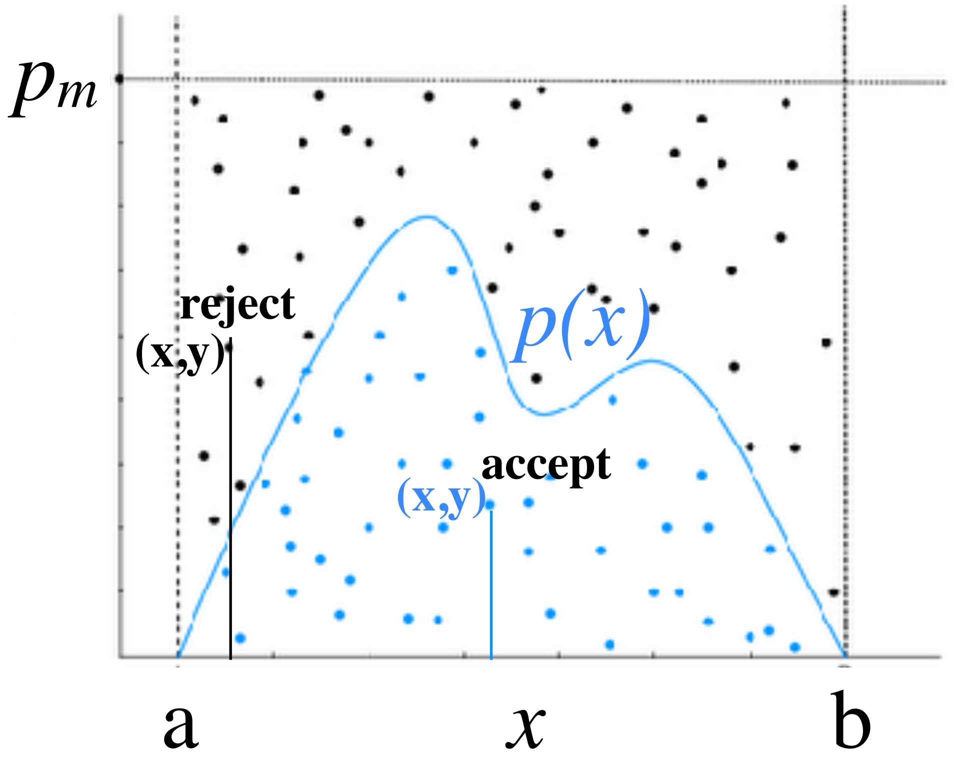


$$x = -\frac{1}{a}\ln(1-y) \quad \text{or (same distribution!)} \quad x = -\frac{1}{a}\ln y$$

Non uniform random numbers distribution: 2) rejection method (general)

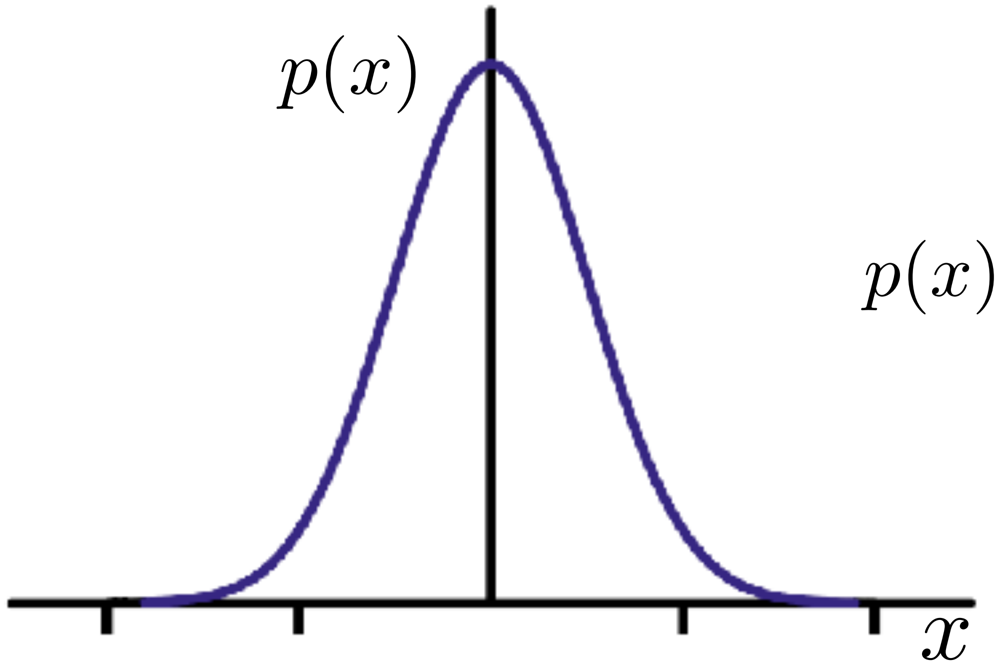
Let $[a,b]$ be the allowed range of values of the variate x, and p_m the maximum of the distribution $p(x)$.

1. Sample a pair of equidistributed random numbers, $x \in [a,b]$ and $y \in [0,p_m]$.
2. If $\boxed{y \leq p(x)}$, accept x as the next random number, otherwise return to step 1.



Due to Von Newmann (1947).
Applicable to almost all distributions.
Can be inefficient if the area of the rectangle $[a,b] \otimes [0,p_m]$ is large compared to the area below the curve p(x)

Non uniform random numbers distribution: 3) gaussian distribution



$$p(x) = \frac{1}{\sigma}\frac{1}{\sqrt{2\pi}}\ e^{-x^2/(2\sigma^2)}$$

How to produce numbers with gaussian distribution?

- Inverse transformation method: impossible

The cumulative distribution function P(x) cannot be analytically calculated!

- Rejection method: inefficient

Non uniform random numbers distribution:
3) gaussian distribution - Box-Muller technique

$$p(x) = \frac{1}{\sigma}\frac{1}{\sqrt{2\pi}}\ e^{-x^2/(2\sigma^2)}$$

Hint: consider the distribution in 2D instead of 1D (here $\sigma = 1$):

$$p(x)p(y)dxdy = (2\pi)^{-1}\ e^{-(x^2+y^2)/2}\ dxdy$$

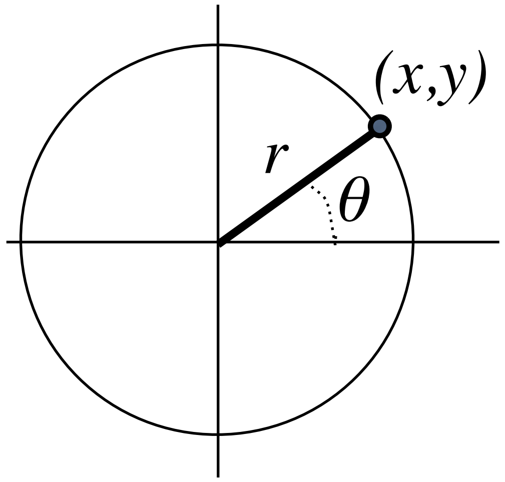
Non uniform random numbers distribution: 3) gaussian distribution - Box-Muller technique

$$p(x) = \frac{1}{\sigma}\frac{1}{\sqrt{2\pi}}\; e^{-x^2/(2\sigma^2)}$$

Hint: consider the distribution in 2D instead of 1D (here $\sigma = 1$):

$$p(x)p(y)dxdy = (2\pi)^{-1}\; e^{-(x^2+y^2)/2}\; dxdy$$

Use polar coordinates: $r = \sqrt{x^2 + y^2}$, $\theta = \arctan{(y/x)}$; def.: $\rho \equiv r^2/2$



Non uniform random numbers distribution: 3) gaussian distribution - Box-Muller technique

$$p(x) = \frac{1}{\sigma}\frac{1}{\sqrt{2\pi}}\; e^{-x^2/(2\sigma^2)}$$

Hint: consider the distribution in 2D instead of 1D (here $\sigma = 1$):

$$p(x)p(y)dxdy = (2\pi)^{-1}\; e^{-(x^2+y^2)/2}\; dxdy$$

Use polar coordinates: $r = \sqrt{x^2+y^2}$, $\theta = \arctan\left(y/x\right)$; def.: $\rho \equiv r^2/2$

$$\rightarrow dxdy = r\; dr\; d\theta = d\rho\; d\theta$$

and therefore:

$$p(x)p(y)\; dx\; dy = p(\rho, \theta)\; d\rho\; d\theta = (2\pi)^{-1}\; e^{-\rho}\; d\rho\; d\theta$$

Non uniform random numbers distribution: 3) gaussian distribution - Box-Muller technique

$$p(x) = \frac{1}{\sigma}\frac{1}{\sqrt{2\pi}}\; e^{-x^2/(2\sigma^2)}$$

Hint: consider the distribution in 2D instead of 1D (here $\sigma = 1$):

$$p(x)p(y)dxdy = (2\pi)^{-1}\; e^{-(x^2+y^2)/2}\; dxdy$$

Use polar coordinates: $r = \sqrt{x^2+y^2}$, $\theta = \arctan(y/x)$; def.: $\rho \equiv r^2/2$

$$\rightarrow dxdy = r\; dr\; d\theta = d\rho\; d\theta$$

and therefore:

$$p(x)p(y)\; dx\; dy = p(\rho,\theta)\; d\rho\; d\theta = \boxed{(2\pi)^{-1}}\; \boxed{e^{-\rho}\; d\rho}\boxed{d\theta}$$

If $\begin{cases} \boxed{\rho} \text{ exponentially distributed} \\ \boxed{\theta} \text{ uniformly distributed in} [0, 2\pi] \end{cases} \rightarrow \begin{cases} x = r\cos\theta = \sqrt{2\rho}\cos\theta \\ y = r\sin\theta = \sqrt{2\rho}\sin\theta \\ x, y \text{ have gaussian distribution} \\ \text{with } \langle x\rangle = \langle y\rangle = 0 \text{ and } \sigma = 1 \end{cases}$

Non uniform random numbers distribution:
3) gaussian distribution - Box-Muller recipe #1

If $\begin{cases} \rho \text{ exponentially distributed} \\ \theta \text{ uniformly distributed in} [0, 2\pi] \end{cases} \rightarrow \begin{cases} x = r\cos\theta = \sqrt{2\rho}\cos\theta \\ y = r\sin\theta = \sqrt{2\rho}\sin\theta \\ x, y \text{ have gaussian distribution} \\ \text{with } \langle x\rangle = \langle y\rangle = 0 \text{ and } \sigma = 1 \end{cases}$

Recipe #1 (BASIC FORM):

$$\begin{cases} X, Y \ \mathit{unif.\ distrib.\ in}\ [0,1[\\ \rho = -\ln(X) \ \mathit{distributed\ with}\ p(\rho) = e^{-\rho} \\ \theta = 2\pi Y \ \mathit{distributed\ with}\ (2\pi)^{-1} p_u \end{cases} \rightarrow \begin{cases} x = r\cos\theta = \sqrt{-2\ln X}\cos(2\pi Y) \\ y = r\sin\theta = \sqrt{-2\ln X}\sin(2\pi Y) \end{cases}$$

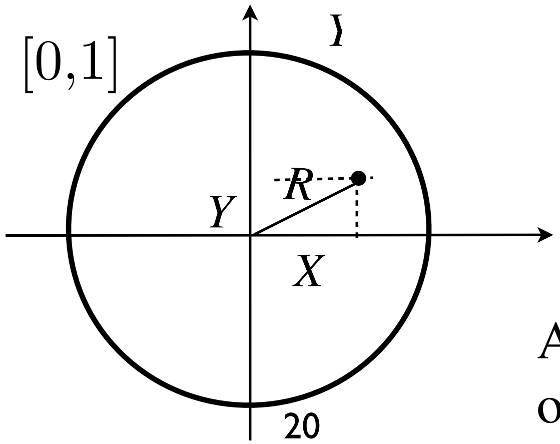
NOTE:
x, y are normally distributed and statistically independent. Gaussian variates with given variances σ_x, σ_y are obtained by multiplying x and y by σ_x and σ_y respectively

Non uniform random numbers distribution: 3) gaussian distribution - Box-Muller recipe #2

If $\begin{cases} \rho \text{ exponentially distributed} \\ \theta \text{ uniformly distributed in}[0,2\pi] \end{cases} \rightarrow \begin{cases} x = r\cos\theta = \sqrt{2\rho}\cos\theta \\ y = r\sin\theta = \sqrt{2\rho}\sin\theta \\ x, y \text{ have gaussian distribution} \\ \text{with } \langle x\rangle = \langle y\rangle = 0 \text{ and } \sigma = 1 \end{cases}$

Recipe #2 (POLAR FORM) (implemented in **boxmuller.f90**) :

$$\begin{cases} X,\ Y \text{ uniformly distributed in } [-1{,}1]; \\ \text{take } (X,Y) \text{ only within the unitary circle;} \\ \Rightarrow R^2 = X^2 + Y^2 \text{ is} \\ \textbf{uniformly} \text{ distributed in } [0{,}1] \end{cases} \rightarrow \begin{cases} x = \sqrt{-2\ln R^2}\,\dfrac{X}{R} \\ y = \sqrt{-2\ln R^2}\,\dfrac{Y}{R} \\ \text{since:} \\ \cos\theta = \dfrac{X}{R},\quad \sin\theta = \dfrac{Y}{R} \end{cases}$$



Advantages: avoids the calculations of sin and cos functions

Some programs:

on moodle2 :

expdev.f90 boxmuller.f90

A look at the expdev.f90 code

```
subroutine expdev(x)
   REAL, intent (out) :: x
   REAL :: r
   do
      call random_number(r)
      if(r > 0) exit
   end do
   x = -log(r)
  END subroutine expdev
```

r is generated in [0,1[;
but r=0 has to be discarded;
if r=0, generate another random number;
if not, exit from the **unbounded** loop
and calculate its log

A look at the boxmuller.f90 code

```
SUBROUTINE gasdev(rnd)
   IMPLICIT NONE
   REAL, INTENT(OUT) :: rnd
   REAL :: r2, x, y
   REAL, SAVE :: g
   LOGICAL, SAVE :: gaus_stored=.false.
   if (gaus_stored) then
     rnd=g
     gaus_stored=.false.
   else
     do
       call random_number(x)
       call random_number(y)
       x=2.*x-1.
       y=2.*y-1.
       r2=x**2+y**2
       if (r2 > 0. .and. r2 < 1.) exit
     end do
     r2=sqrt(-2.*log(r2)/r2)
     rnd=x*r2
     g=y*r2
     gaus_stored=.true.
   end if
END SUBROUTINE gasdev
```

Every two calls
uses the random number
already generated in the previous call

2 examples of optimization!

$\longrightarrow$ since:

$$x = \sqrt{-2\ln R^2}\,\frac{X}{R} = X\sqrt{-2\ln R^2/R^2}$$

(thus avoiding the calculation of
another $\sqrt{\ }$ to calculate R separately)

A look at the gasdev.c code

```
#include <math.h>

float gasdev(long *idum)
{
    float ran1(long *idum);
    static int iset=0;
    static double gset;
    double fac,rsq,v1,v2;

    if (iset == 0) {
        do {
            v1=2.0*ran1(idum)-1.0;
            v2=2.0*ran1(idum)-1.0;
            rsq=v1*v1+v2*v2;
        } while (rsq >= 1.0 || rsq == 0.0);
        fac=sqrt(-2.0*log(rsq)/rsq);
        gset=v1*fac;
        iset=1;
        return (float)(v2*fac);
    } else {
        iset=0;
        return (float)gset;
    }
}
```

Every two calls
uses the random number
already generated in the previous call

2 examples of optimization!

$\rightarrow$ since: $x = \sqrt{-2\ln R^2}\,\frac{X}{R} = X\sqrt{-2\ln R^2/R^2}$

(thus avoiding the calculation of another $\sqrt{\ }$ to calculate R separately)

Other programs:

(optional, but useful!)

random.f90 (is a module)
t_random.f90

to compile:
$gfortran random.f90 t_random.f90
(the module first!)

Part II -
Using random numbers to simulate random processes

Random processes: radioactive decay

$N(t)$	Atoms present at time t
λ	Probability for each atom to decay in Δt
$\Delta N(t)$	Atoms which decay between t and $t + \Delta t$

$$\Delta N(t) = -\lambda N(t) \Delta t$$

we use the probability λ of decay of each atom to simulate the behavior of the number of atoms left; we should be able to obtain (on average):

$$N(t) = N(t = 0) e^{-\lambda t}$$

Radioactive decay: numerical simulation

A scheme for the simulation

1. Assign a value to the decay constant $\lambda \leq 1$ (the probability for each nucleus to decay in a given interval of time Δt)

   λ establishes the time scale; one iteration in the "do loop" corresponds to one time step Δt

2. Start with **Nleft** $=$ **Nstart**$=$ total number of nuclei at time $t = 0$

3. Basic algorithm: **for each nucleus** left (not yet decayed):

   - Generates a random number $0 \leq x \leq 1$
   - if $x \leq \lambda$, the nucleus decays and **Nleft** $=$ **Nleft - 1**, otherwise it remains and **Nleft** is unchanged.

4. Repeat for each nucleus

5. Repeat the cycle for the next time step

Note: unbounded loop

```
DO                                      ! loop on time
  DO  i = 1, nleft                      ! loop on all the nuclei left
    call random_number(r)
    IF (r <= lambda) THEN               ! BASIC   ALGORITHM
      nleft = nleft - 1                 !  update the nuclei left (*)
    ENDIF
  END DO
WRITE (unit=7,fmt=*) t , nleft
  if (nleft == 0) exit
  t = t + 1
END DO
```

Note: "exit" $\neq$ "cycle"

(*) Notice that the upper bound of the inner loop (nleft) is changed within the execution of the loop; but with most compilers, in the execution the **loop** goes on up to the **initial value of the upper bound** (nleft); this ensures that the implementation of the algorithm is correct. The program checkloop.f90 is a test for the behavior of the loop. Look also at decay_checkloop.f90. If nleft would be changed (decreased) during the execution, the effect would be an overestimate of the decay rate. CHECK with your compiler!

Programs:

decay.f90
decay_checkloop.f90

checkloop.f90

Details on Fortran: unbounded loops

```
[name:] DO
        exit [name]

or [name:] DO
           END DO [name]
```

(name is useful in case of nested loops for explicitly indicating which loop we exit from)

Alternative form: ”do while” loop

Always set a condition to exit from a loop! E.g.:

```
DO
 if (condition)exit
END DO
```

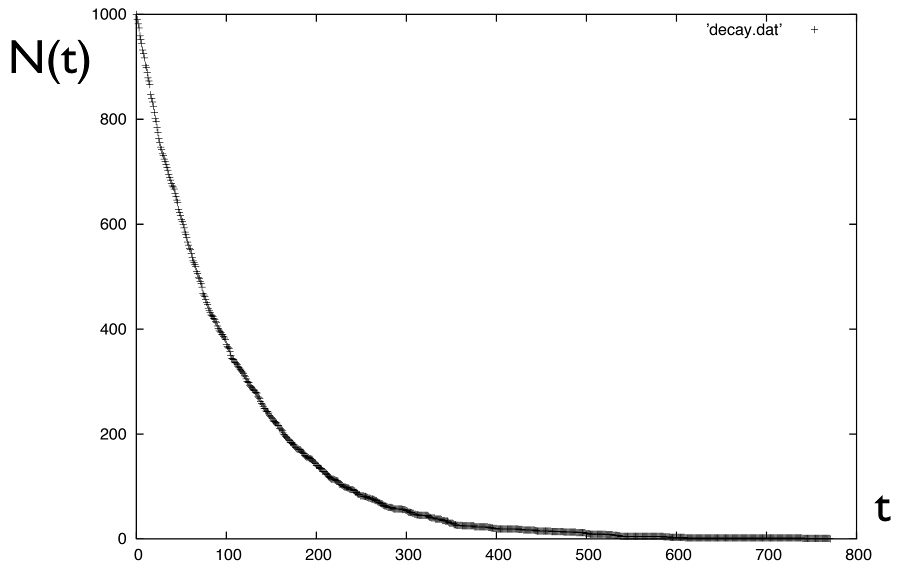
or:

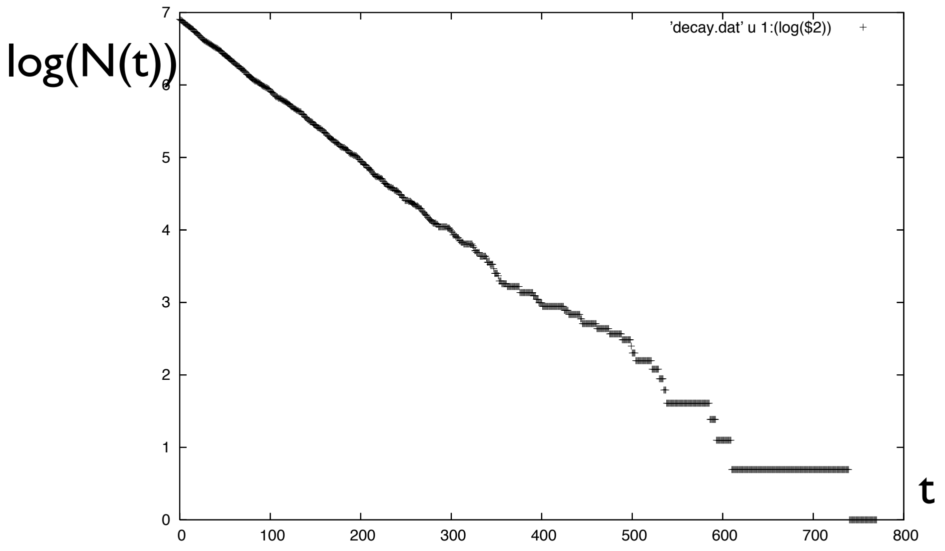
```
DO WHILE (.not. condition)
 ...
END DO
```

NOTE: first is better (“if () ..exit” can be placed everywhere in the loop,
whereas DO WHILE must execute the loop up to the end)

- Additional note:
Difference between EXIT and CYCLE

Radioactive decay:
results of numerical simulation





results of decay simulation (N vs t) with N=1000

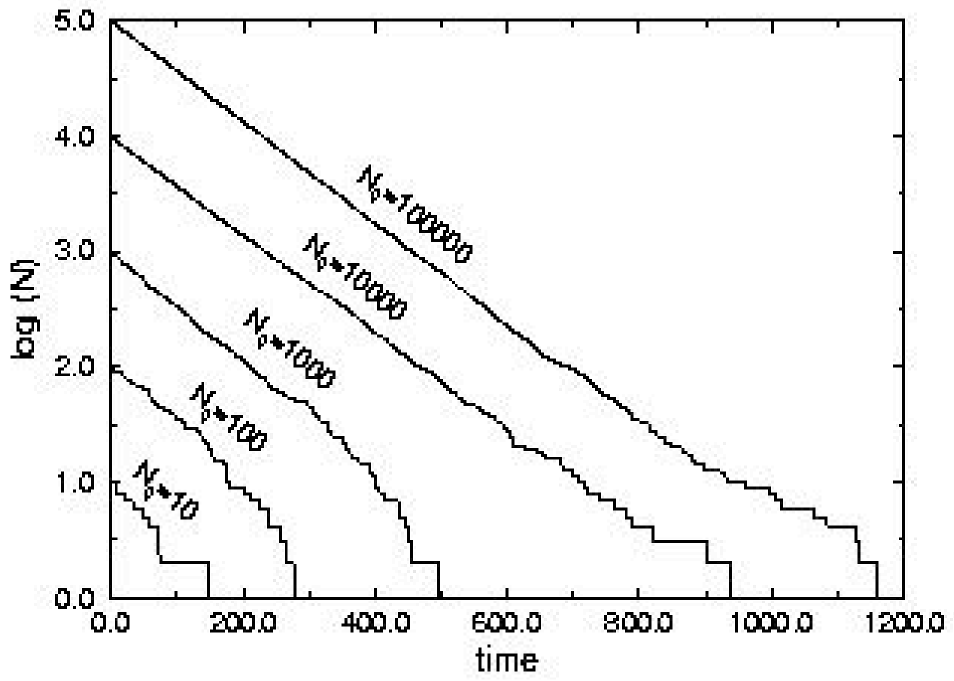
$N(t) \sim N_0 \exp(-a\,t)$

semilog plot (log(N) vs t)

=> $\log(N(t)) = \log N_0 - a\,t$

=> slope is $-a$

Radioactive decay:
results of numerical simulation



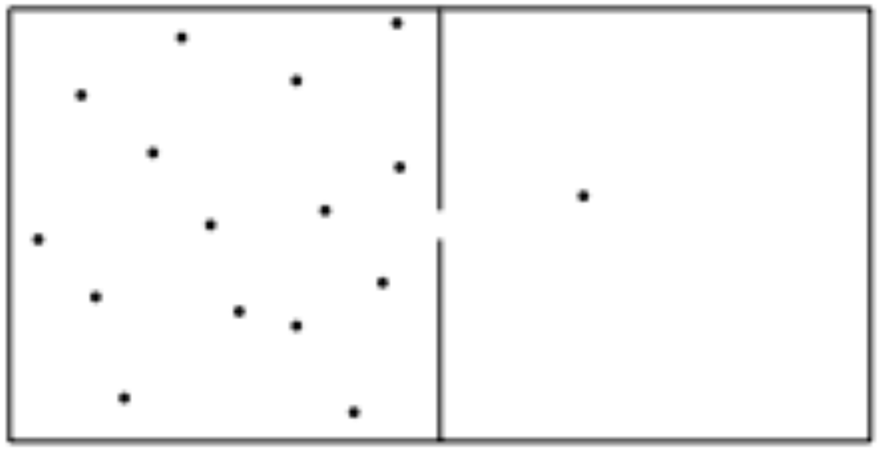
Semilog plots of the results of simulations for the same decay rate and different initial number of atoms:
almost a straight line, but with important deviations (stochastic) for small N

Stochastic simulations give reliable results when obtained:

- on average and for large numbers
- fine discretisation of time evolution

(homework: change λ; compare the value obtained from the simulation with the one inserted; does the "quality" of the results change with λ?)

Other random processes: order and disorder



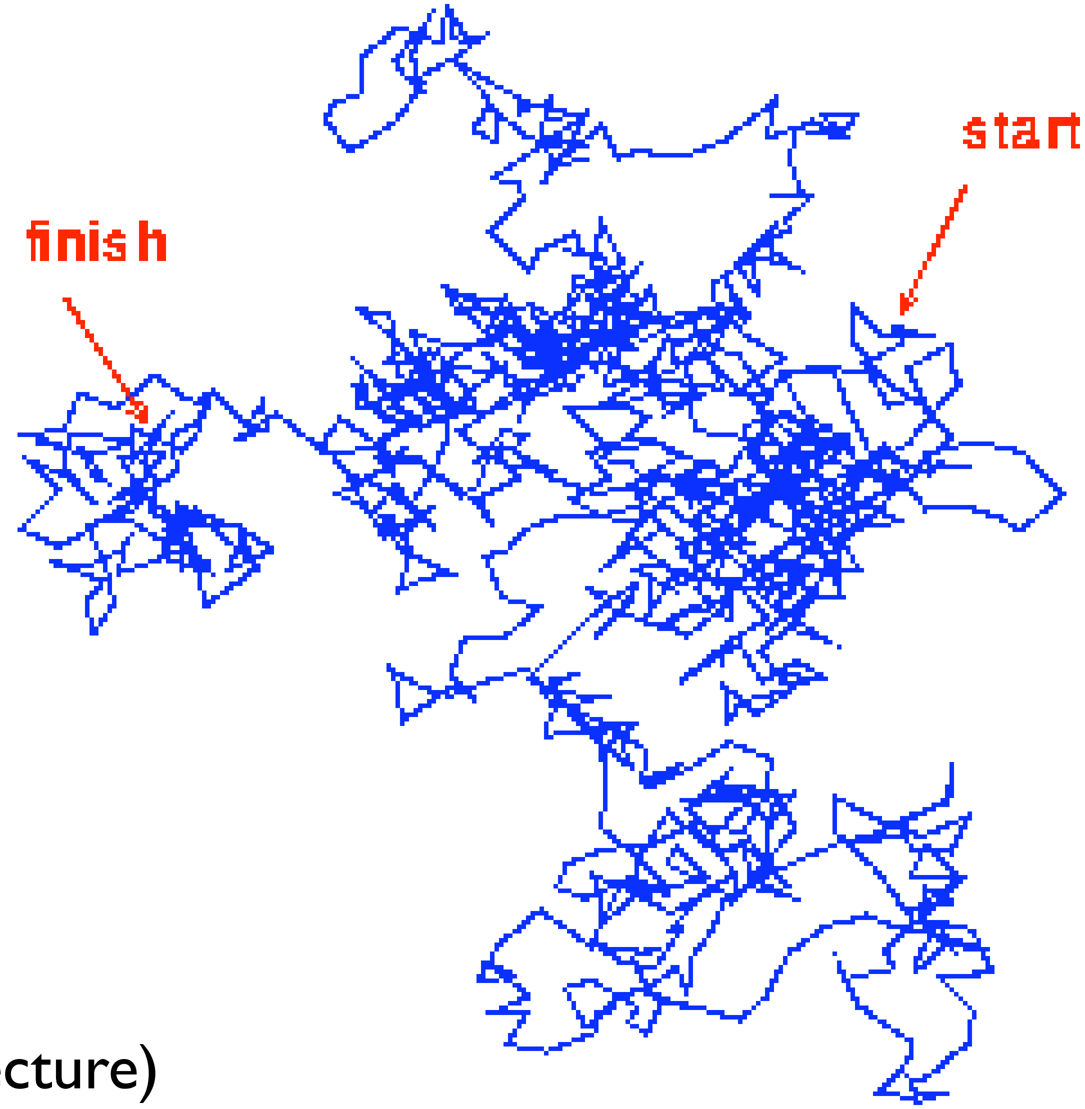
A box is divided into two parts communicating through a small hole. One particle randomly can pass through the hole per unit time, from the left to the right or viceversa.

$N_{left}(t)$: number of particles present at time t in the left side

Given $N_{left}(0)$, what is $N_{left}(t)$?

(more on that in a future Lecture)

Other random processes: random walks



(see next lecture)

Part III - Fitting data

Least-square method

- Suppose to have N_D data (independent measurements of the variable y which is function of the variable x):

$$(x_i, y_i \pm \sigma_i), \qquad i = 1, N_D$$

with $\pm\sigma_i$ error associated to the i value of y.

- Purpose: determine the function $y = f(x)$ which better described these data. If the analytic form of the function is known, a part from a set M_P of parameters $\{a_1, a_2 \ldots, a_{M_P}\}$, i.e., $f(x) = f(x; \{a_m\})$, the goal is to find the best set of parameters.

- To test whether the data *fit* via $f(x)$ is good or not calculate the quantity

$$\chi^2 := \sum_{i=1}^{N_D} \left(\frac{y_i - f(x_i; \{a_m\})}{\sigma_i} \right)^2$$

  Note that by dividing by σ_i, data with larger errors have smaller weight in this weighted average.

- The smallest χ^2, the better the fit is.

- **Least-squares fitting**: The parameters M_P that minimize χ^2 are found by:

$$\frac{\partial \chi^2}{\partial a_m} = 0 \quad (m = 1, M_P)$$

$$\Longrightarrow \sum_{i=1}^{N_D} \frac{y_i - f(x_i)}{\sigma_i^2} \frac{\partial f(x;\{a_m\})}{\partial a_m} = 0 \qquad (1)$$

example: see program **fit.f90**

- If $f(x;a,b) = ax+b$ **(linear regression)**, the equations giving χ^2 minimum reduce to:

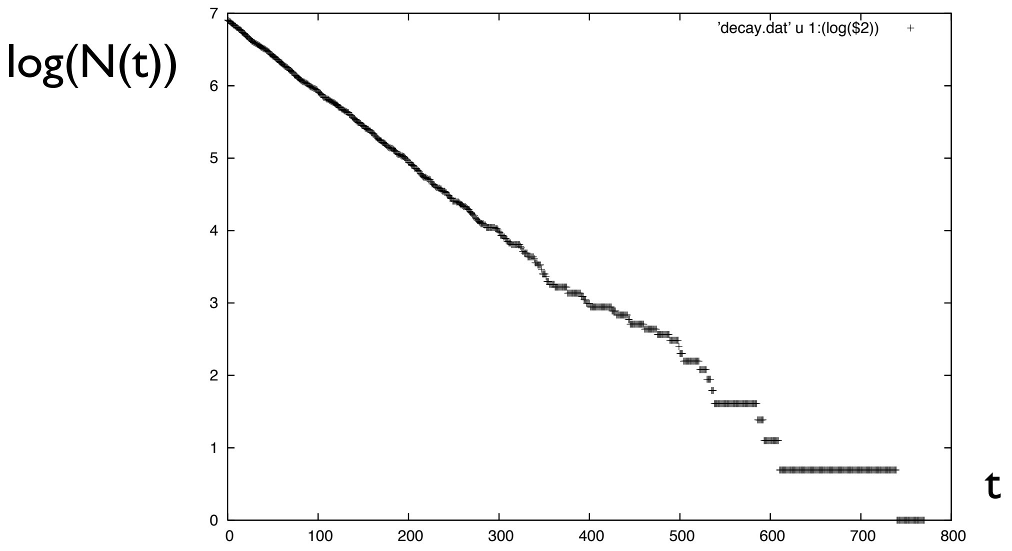
$$a = \frac{SS_{xy} - S_x S_y}{\Delta}, \quad b = \frac{S_{xx} S_y - S_x S_{xy}}{\Delta}$$

$$S = \sum_{i=1}^{N_D} \frac{1}{\sigma_i^2}, \qquad S_x = \sum_{i=1}^{N_D} \frac{x_i}{\sigma_i^2}$$

$$S_y = \sum_{i=1}^{N_D} \frac{y_i}{\sigma_i^2}, \qquad S_{xx} = \sum_{i=1}^{N_D} \frac{x_i^2}{\sigma_i^2}$$

$$S_{xy} = \sum_{i=1}^{N_D} \frac{x_i y_i}{\sigma_i^2}, \quad \Delta = SS_{xx} - S_x^2 \qquad (2)$$

Examples - linear regression

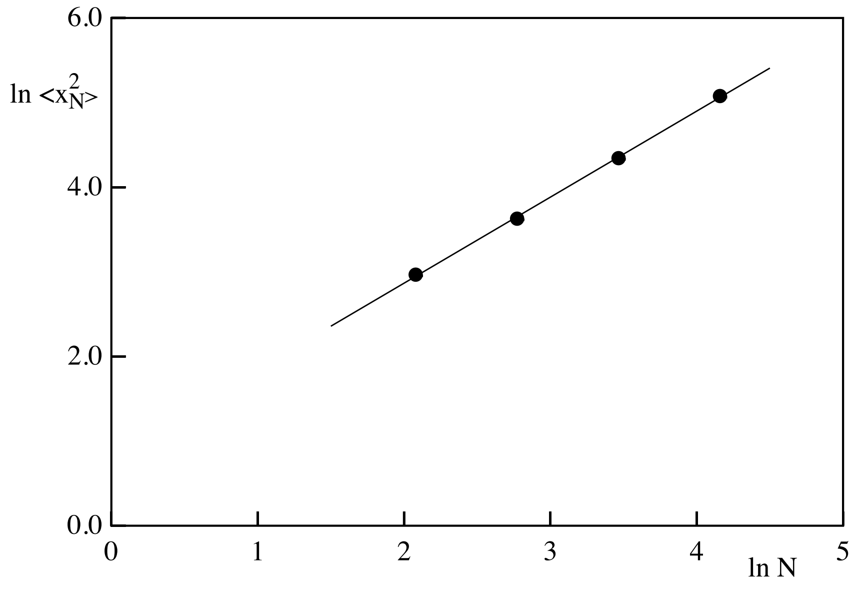


radioactive decay:

$N(t) \sim N_0 \exp(- a\, t)$

we can fit with the exp.
but it is better to fit:

$\log(N(t)) = \log N_0 - a\, t$



Random walk:

$\langle x^2_N \rangle \sim N^a$

but it is better to fit:

$\log \langle x^2_N \rangle = a \log N$

Example: fit using gnuplot - 1

Suppose you want to fit your data (say, ‘data.dat’) with an exponential function.
You have to give: 1) the functional form ; 2) the name of the parameters

```
gnuplot> f(x) = a * exp (-x*b)
```

Then we have to recall these informations together with the data we want to fit:
it can be convenient to inizialize the parameters:

```
gnuplot> a=0. ; b=1.
```
(for example)

```
gnuplot> fit f(x) 'data.dat' via a,b
```

On the screen you will have something like:

```
Final set of parameters Asymptotic Standard Error
======================= =========================

a = 1 +/- 8.276e-08 (8.276e-06%)
b = 10 +/- 1.23e-06 (1.23e-05%)

correlation matrix of the fit parameters:

a b
a 1.000
b 0.671 1.000
```

It’s convenient to plot together the original data and the fit:

```
gnuplot> plot f(x), 'data.dat'
```

Example: fit using gnuplot - II

If you prefer to use linear regression, **use logarithmic data in the data file**, **or** directly fit the log of the original data using **gnuplot:**

```
gnuplot> f(x) = a + b*x
```

Then we have to recall these informations together with the data we want to fit
(in the following example: x=log of the first column; y=log of the second column):

```
gnuplot> fit f(x) 'data.dat' u (log($1)):(log($2)) via a,b
```

```
...
  Final set of parameters Asymptotic Standard Error
  ======================= ==========================  (...gnuplot will work for you....)
...
```

Also in this case it will be convenient to plot together the original data and the fit:

```
gnuplot> plot f(x), 'data.dat' u (log($1)):(log($2))
```

In case of needs, we can limit the set of data to fit in a certain range **[x_min:x_max]**:

```
gnuplot> fit [x_min:x_max] f(x) 'data.dat' u ... via ...
```

Part IV -
more on fortran

A few notes on Fortran

related to the exercises

Intrinsic functions:

LOGARITHM

log returns the natural logarithm

log10 returns the common (base 10) logarithm

(NOTE: also in **gnuplot**, **log** and **log10** are defined with the same meaning)

INTEGER PART

nint(x) and the others, similar but different (see Lect. II) => ex. II requires histogram for negative and positive data values

Arrays:

possible to label the elements from a negative number or 0:

dimension array(-n:m) (e.g., useful for making histograms)

[default in Fortran: n=1; in c and c++: n=0]

Array dimension:

default : dimension array([1:]n)

but also using other dimensions e.g.:   dimension array(-n:m)

Important to **check dimensions** of the array when compiling or during execution !

If not done, it is difficult to interpret error messages (typically: “segmentation fault”),   or even possible to obtain unpredictable results!

Default in g95 and gfortran:

boundaries not checked; use **compiler option**:

gfortran -fcheck=bounds myprogram.f90

(obsolete but still active alternative: -fbounds-check)

Typing (Unix line command):

man gfortran

you can scroll the manual pages and see the possible compilation options

Some Fortran compiler options

-fcheck=bounds   enables checking for array subscript expressions

-fbacktrace   generate extra information to provide source file traceback at run time
Specify that, when a runtime error is encountered or a deadly signal is emitted (segmentation fault, illegal instruction, bus error or floating-point exception), the Fortran runtime library should output a backtrace of the error. This option only has influence for compilation of the Fortran main program.

-Wall   Enables commonly used warning options

…

Making histograms: use int() or similar intrinsic functions?

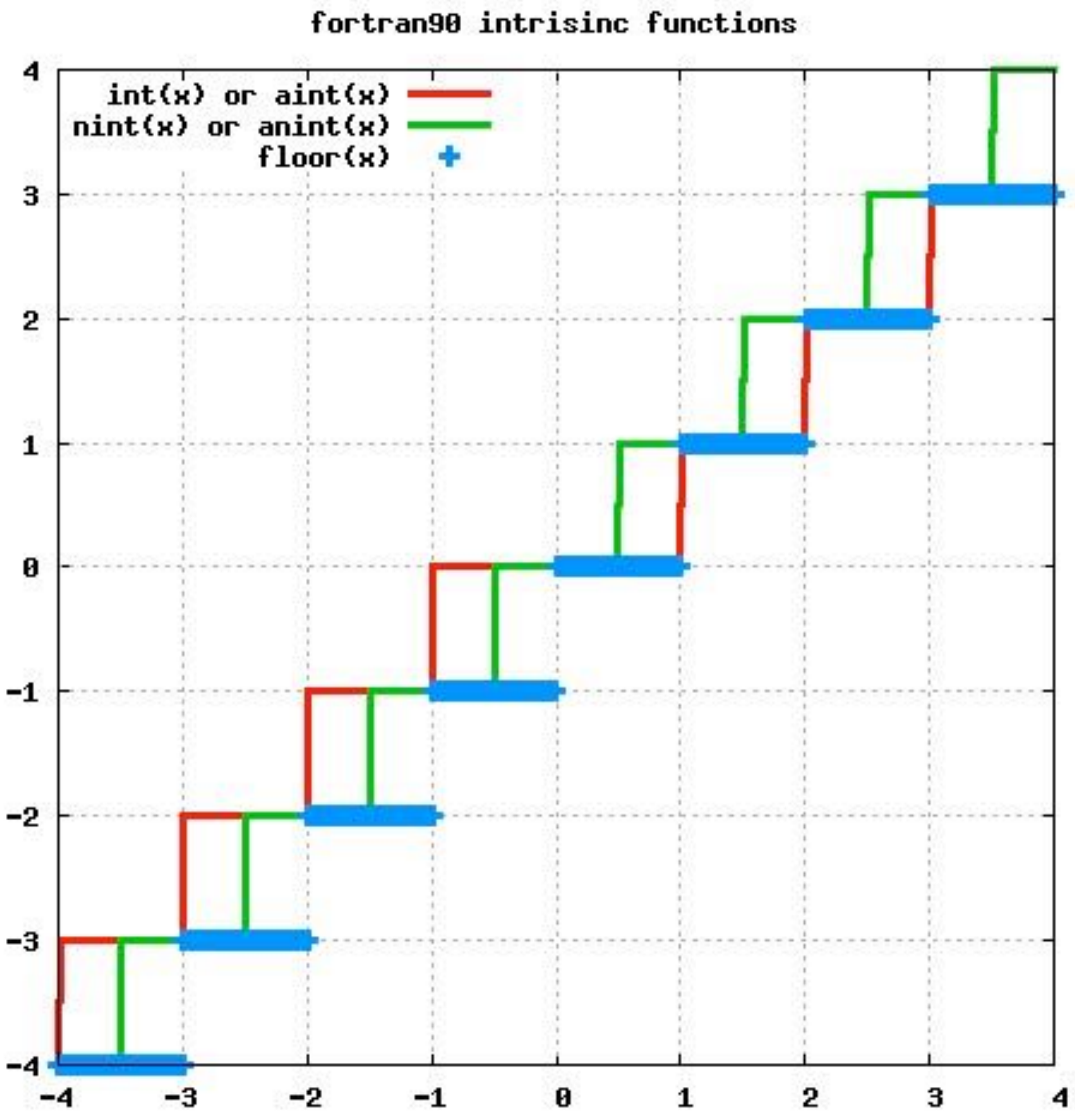
AINT(A[,KIND])
· Real elemental function
· Returns A truncated to a whole number. AINT(A) is the largest integer which is smaller than |A|, with the sign of A. For example, AINT(3.7) is 3.0, and AINT(-3.7) is -3.0.
· Argument A is Real; optional argument KIND is Integer

ANINT(A[,KIND])
· Real elemental function
· Returns the nearest whole number to A. For example, ANINT(3.7) is 4.0, and AINT(-3.7) is -4.0.
· Argument A is Real; optional argument KIND is Integer

FLOOR(A,KIND)
· Integer elemental function
· Returns the largest integer ≤ A. For example, FLOOR(3.7) is 3, and FLOOR(-3.7) is -4.
· Argument A is Real of any kind; optional argument KIND is Integer
· Argument KIND is only available in Fortran 95

INT(A[,KIND])
· Integer elemental function
· This function truncates A and converts it into an integer. If A is complex, only the real part is converted. If A is integer, this function changes the kind only.
· A is numeric; optional argument KIND is Integer.

NINT(A[,KIND])
· Integer elemental function
· Returns the nearest integer to the real value A.
· A is Real


fortran90 intrisinc functions
int(x) or aint(x)
nint(x) or anint(x)
floor(x)
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4

Structure of a main program with one function

(see: expdev.f90 or boxmuller.f90)

```
program name_program
  implicit none  (*)
 <declaration of variables>
 <executable statements>

contains
   subroutine ...  (or function)
    ...
    end subroutine

end program
```

(*) General suggestion for variable declaration:
Use "implicit none" + explicit declaration of variables

See also the use of **module** in previous Lectures