# 993SM - Laboratory of Computational Physics week 7 November 4, 2024

**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

# Monte Carlo methods

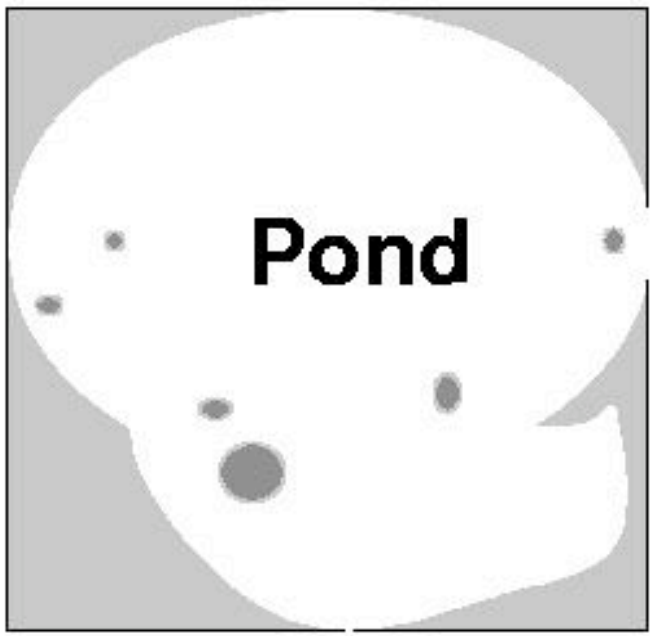
- Monte Carlo integration
- Metropolis method to generate non-uniform random number distributions

# Monte Carlo methods:

## "acceptance-rejection" or "hit or miss"

**(to calculate areas)**

which is $A_{pond}$ ?




- enclose the pond in a box of Area $A_{box}$
- throw pebbles uniformly and randomly in the box
- count the number of pebbles felt in the pond with respect to the number felt in the box
- Assuming a uniform distribution, the number of pebbles falling into the ponds is proportional to the area of the pond:

$$\frac{N_{pond}}{N_{pond}+N_{box}} = \frac{A_{pond}}{A_{box}}$$

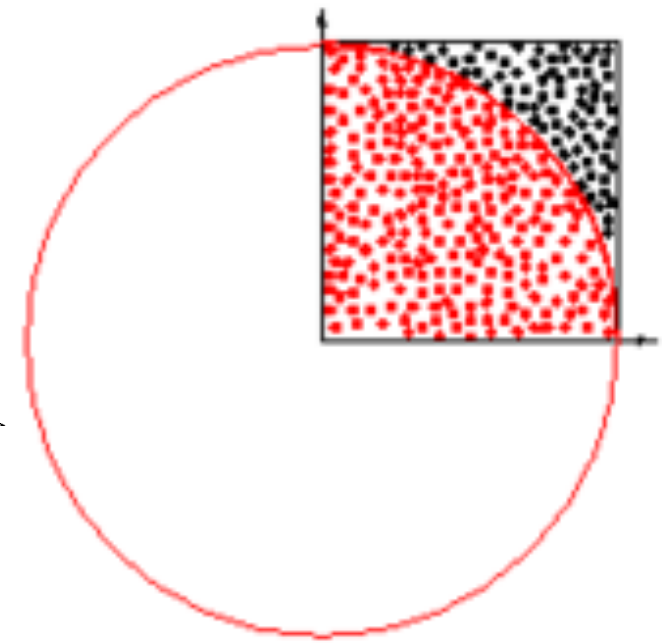
$$\Rightarrow \quad A_{pond} = \frac{N_{pond}}{N_{pond}+N_{box}} A_{box}$$

# Monte Carlo methods:

## "acceptance-rejection" or "hit or miss"

**(to calculate areas)**

$$\pi = ???$$



$N$ random points in the unit square
coordinates $x_i, y_i$

Then, the number of
points $N_c$ lying within the quarter circle (i.e. fulfilling the relation $x^2 + y^2 \leq 1$) is compared to the total number $N$ of points and the fraction will give us an approximate value of $\pi$:

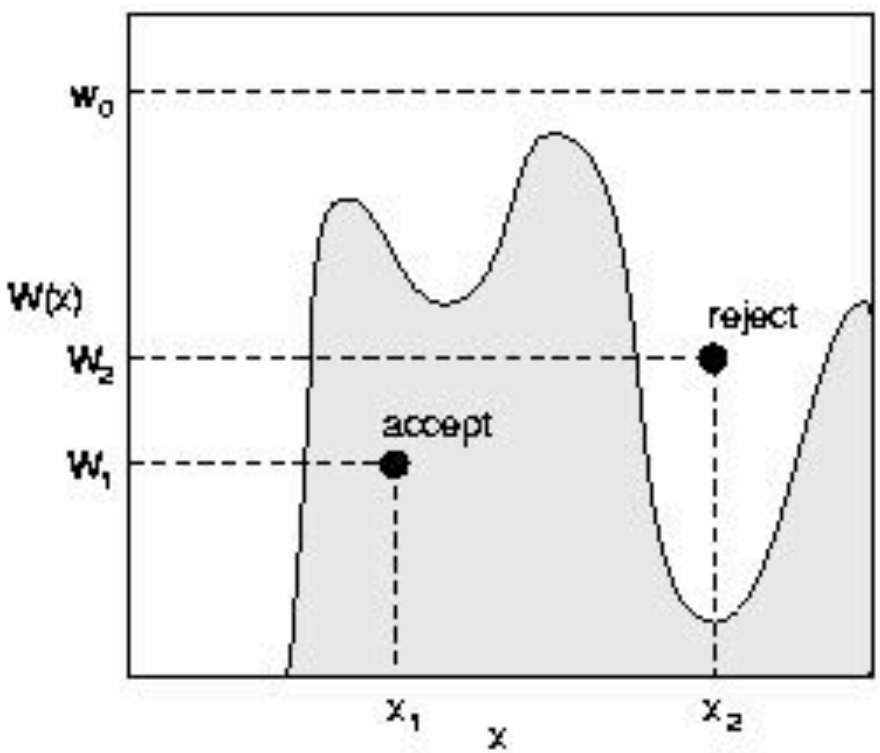
$$\pi(N) = 4\frac{N_c(N)}{N}$$

# Monte Carlo methods:

## "acceptance-rejection" or "hit or miss"

**(to calculate definite integrals)**

$$\int W(x)dx = \ ?$$



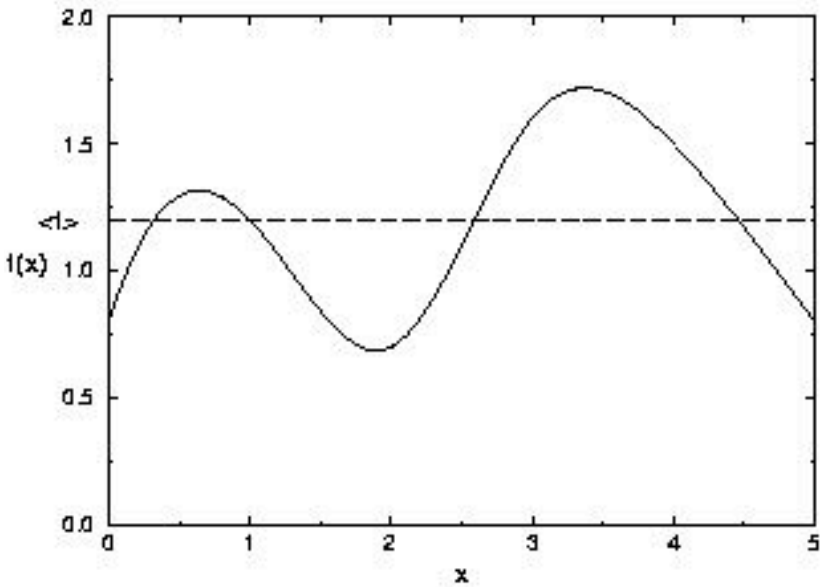

For W(x) positive in the integration interval, the value of the area under W(x) can be obtained by producing random points (i.e. (x,y) random pairs) uniformly distributed in a rectangle containing W(x).

For each point (x,y) compare y with W(x): if y<W(x), the point is accepted. The area under W(x) is the number of points accepted divided by the total number of points generated and multiplied by the area of the rectangle.

(remember: also used to generate random numbers $x_i$ distributed according W(x))

# Other simple
# Monte Carlo methods

We can always write: $$I = \int_a^b f(x)dx = (b-a)\langle f \rangle$$

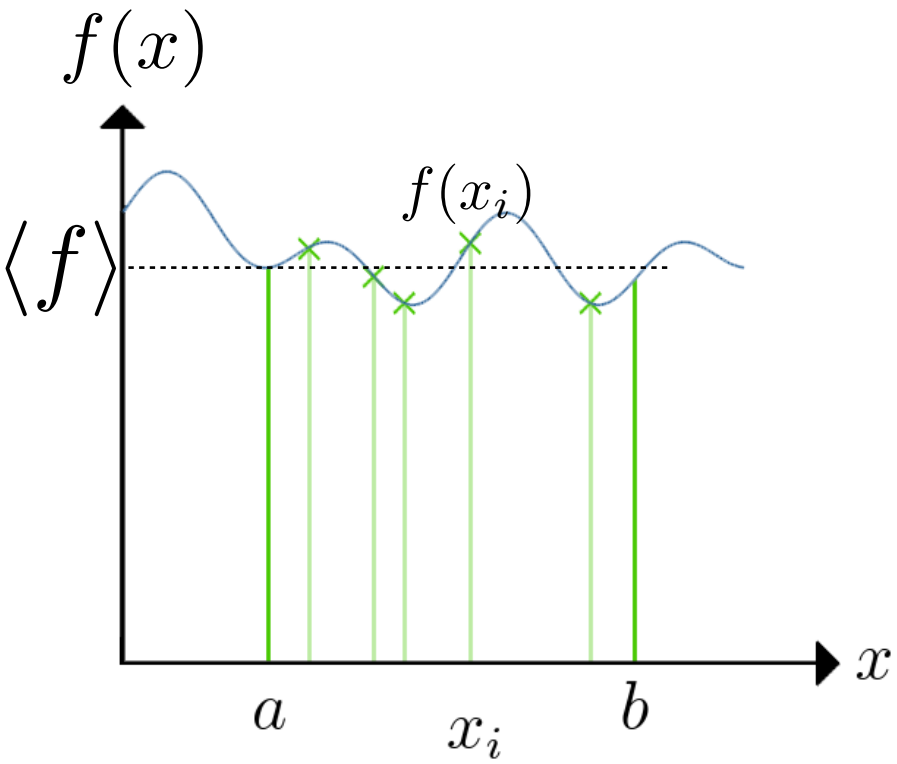



i.e., the value of the integral of f(x) between a and b equals the length of the interval (b-a) times the average value of the function <f> over the same interval.

(If f:[a,b] → R is a continuous function, then there exists a number c in [a,b] such that f(c)=<f>
**(mean value theorem for integration))**

**how to estimate <f> efficiently and accurately?**

# A simple Monte Carlo method: "sample mean"




$$I = \int_a^b f(x)dx = (b-a)\langle f \rangle$$

The sample mean can be calculated by sampling the function *(if smooth enough...)* with a sequence of N uniform random numbers in [a,b]:

$$\langle f \rangle \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

$$\int_a^b f(x)dx \approx (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i) = (b-a)\langle f \rangle$$

# Monte Carlo methods:

## error estimate

Example: MC estimate of π (exact value known)

We can use either acceptance-rejection or sample mean method: $I = 4\int_0^1 \sqrt{1-x^2} = \pi = 3.1416\ldots$

Since we know the "exact" result *I*, we can calculate the **error** in two ways:

1) the **actual error** from the difference with respect to the exact value:

$\Delta_n = |F_n - I|$ with $F_n = (b-a)\frac{1}{n}\sum_{i=1}^{n} f(x_i), \quad x_i \text{ random}$

2) the numerical error from the **variance of the data** $\{f(x_i)\}$:

$\sigma^2 = \langle f^2 \rangle - \langle f \rangle^2$

where

$\langle f \rangle = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$ and $\langle f^2 \rangle = \frac{1}{n}\sum_{i=1}^{n} f(x_i)^2$

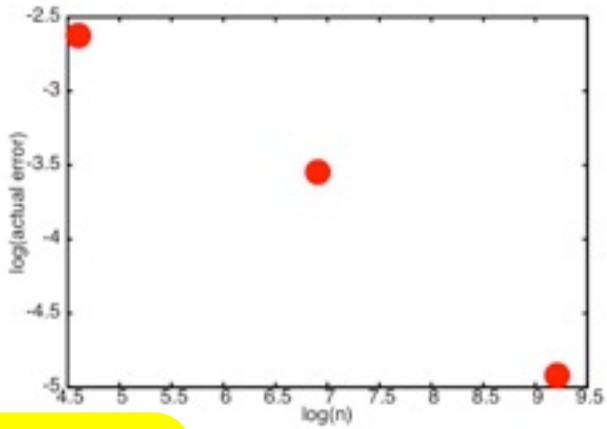
# Monte Carlo methods:

## error estimate

Results:

$$I = 4\int_0^1 \sqrt{1-x^2} = \pi = 3.1416\ldots$$

| $n$ | $F_n$ | actual error | $\sigma_n$ |
|---|---|---|---|
| $10^2$ | 3.0692 | 0.0724 | 0.8550 |
| $10^3$ | 3.1704 | 0.0288 | 0.8790 |
| $10^4$ | 3.1489 | 0.0073 | 0.8850 |




1) the actual error $\Delta_n$ decreases as $1/n^{1/2}$

2) the numerical error from the variance of the data, $\sigma_n$, is roughly constant and is much larger than the actual error

**what is the correct error estimate?**

# Monte Carlo methods:
## error estimate

...typically you do not know which is the "actual error" (you do not know the "true" value and you cannot compare your result with that!)....
but we would like to give an error to our numerical estimate...
(to which extent is our numerical estimate reliable?)

Two methods to estimate the error numerically from the variance of the data (**"reduction of variance"**):

I) average of the averages

II) block average

# MC error handling: method I
# "average of the averages"

make additional runs of $n$ trials each.
Let $M_\alpha$ be the average of each run :

| run $\alpha$ | $M_\alpha$ | actual error |
|---|---|---|
| 1 | 3.1489 | 0.0073 |
| 2 | 3.1326 | 0.0090 |
| 3 | 3.1404 | 0.0012 |
| 4 | 3.1460 | 0.0044 |
| 5 | 3.1526 | 0.0110 |
| 6 | 3.1397 | 0.0019 |
| 7 | 3.1311 | 0.0105 |
| 8 | 3.1358 | 0.0058 |
| 9 | 3.1344 | 0.0072 |
| 10 | 3.1405 | 0.0011 |

$one\ run \equiv n = 10^4\ trials\ each$

Examples of Monte Carlo measurements of the mean value of $f(x) = 4\sqrt{1-x^2}$ in the interval $[0, 1]$. A total of 10 measurements of $n = 10^4$ trials each were made. The mean value $M_\alpha$ and the actual error $|M_\alpha - \pi|$ for each measurement are shown.

Calculate: $\sigma_m{}^2 = \langle M^2 \rangle - \langle M \rangle^2$ with $\langle M \rangle = \frac{1}{m}\sum_{\alpha=1}^{m} M_\alpha$, $\langle M^2 \rangle = \frac{1}{m}\sum_{\alpha=1}^{m} M_\alpha{}^2$

$\Longrightarrow \sigma_m = 0.0068$

$\sigma_m$ is consistent with the results for the actual errors

# MC error handling: method II "block averages"

Instead of doing additional measurements, divide them into "s SUBSETS" and let $S_k$ be the average within each subset :

| subset $k$ | $S_k$ |
|---|---|
| 1 | 3.14326 |
| 2 | 3.15633 |
| 3 | 3.10940 |
| 4 | 3.15337 |
| 5 | 3.15352 |
| 6 | 3.11506 |
| 7 | 3.17989 |
| 8 | 3.12398 |
| 9 | 3.17565 |
| 10 | 3.17878 |

The variance associated to the average of the subsets ${\sigma_s}^2 = \langle S^2 \rangle - \langle S \rangle^2$ gives $\sigma_s = 0.025$, but

$\sigma_s/\sqrt{s}$, which for our example is approximately $0.025/\sqrt{(10)} \approx 0.008$.

is consistent with the actual error

# Monte Carlo methods:

## error estimate - variance reduction

## summary

$$\sigma_n/\sqrt{n} \approx \sigma_m \approx \sigma_s/\sqrt{s}$$

from the variance of the whole set of data

Note: for uncorrelated data !

(proof)

the variance of the **average of the averages**

from the variance of the **block averages**

the most convenient!
but: change block size
and check that
it does not change

# Monte Carlo methods:
## summary

We have introduced :

* "acceptance-rejection"

* "sample mean" to estimate $\langle f \rangle \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$

both OK for smoothly varying functions, but
not very efficient for rapidly varying functions

**How to improve the efficiency of MC integration?**

# A trick for numerical integration: "reduction of variance"

(Note: same word, but different meaning w.r.t. previous slides on error handling)

Given a function $f(x)$ to integrate, suppose that $g(x)$ exists, whose integral is known and such that:

$$|f(x) - g(x)| << \varepsilon$$

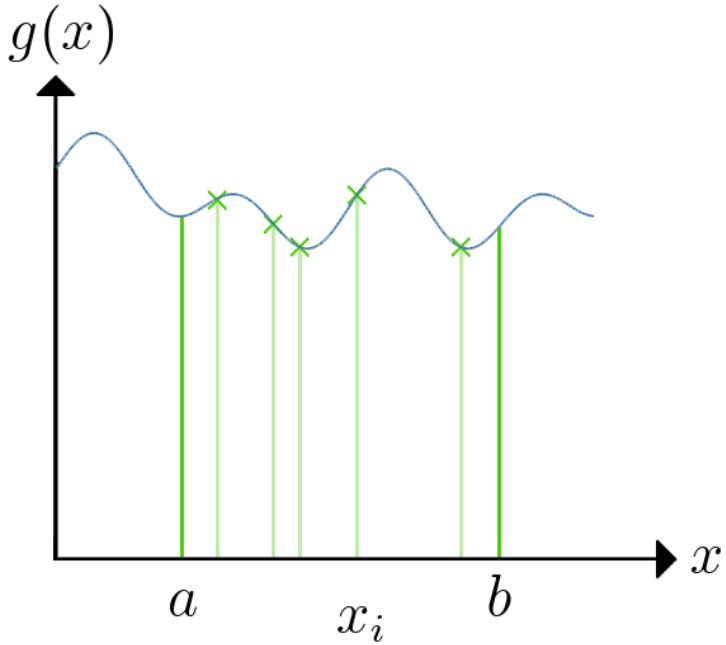
Therefore:

$$F = \int_a^b f(x)dx = \int_a^b \big((f(x)-g(x))+g(x)\big)dx = \int (f(x) - g(x))\, dx + \int g(x)dx$$

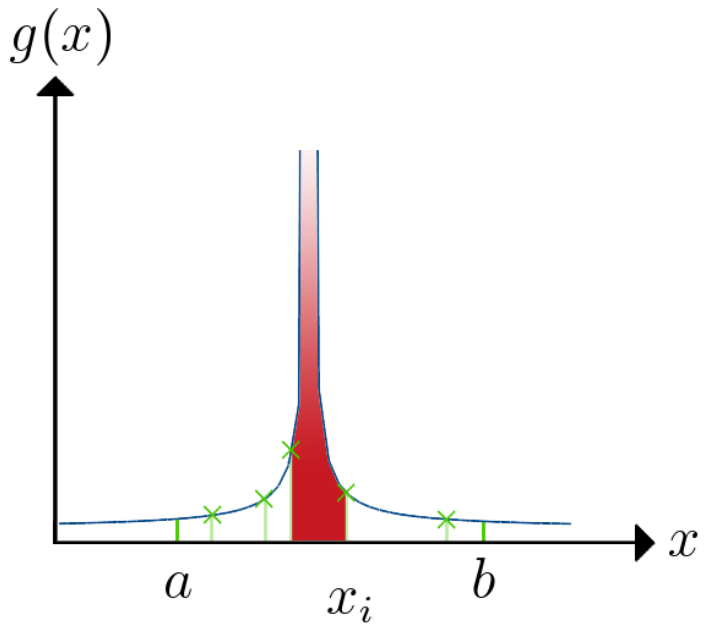
easy to calculate

# Another simple Monte Carlo method: "importance sampling"

Mean value: easy to calculate for smoothly varying functions.
But not for functions rapidly varying.




smooth function




function with singularity

**How to manage such cases?**

# Another simple Monte Carlo method: "importance sampling"

Mean value: easy to calculate for smoothly varying functions.
Idea: in order to calculate:

$$\langle f \rangle \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

consider a distribution function $p(x)$ easy to integrate analytically and close to $f(x)$:

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

where $\left\langle \frac{f(x)}{p(x)} \right\rangle \approx \frac{1}{N} \sum_{i=1}^{N} \left[\frac{f(x_i)}{p(x_i)}\right]$

(particular case: uniform distrib. p(x)=1/(b-a) ...)

with $\{x_i\}$ distributed according to $p(x)$

# Monte Carlo methods:
## "importance sampling"

Calculate:

$$F = \int_0^1 e^{-x^2}\, dx.$$

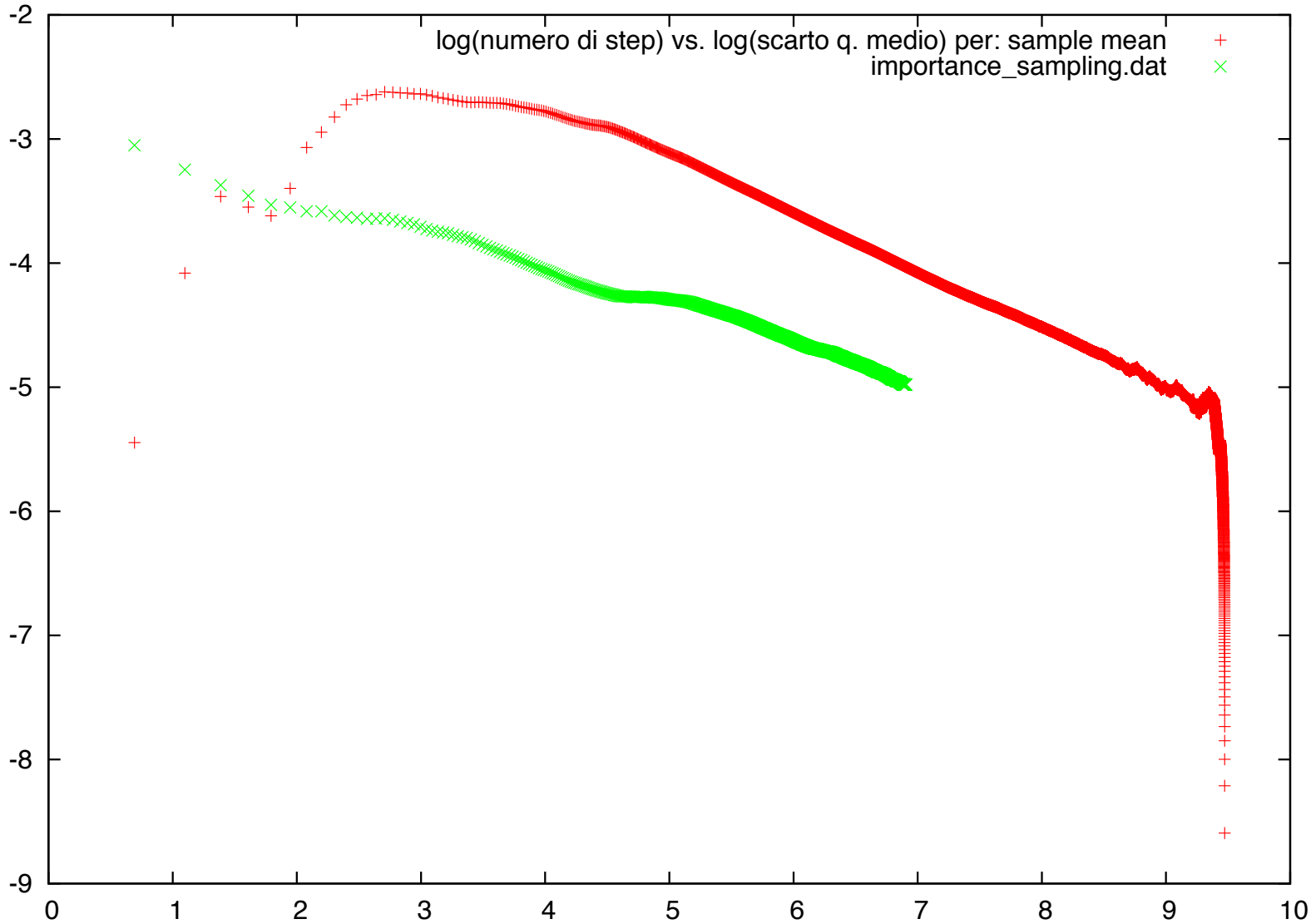
with "sample mean" with random numbers with uniform distribution or using the "importance sampling" with $p(x) = e^{-x}$

| | $p(x) = 1$ | $p(x) = Ae^{-x}$ |
|---|---|---|
| $n$ (trials) | $4 \times 10^5$ | $8 \times 10^3$ |
| $F_n$ | 0.7471 | 0.7469 |
| $\sigma$ | 0.2010 | 0.0550 |
| $\sigma/\sqrt{n}$ | $3 \times 10^{-4}$ | $6 \times 10^{-4}$ |
| Total CPU time (s) | 35 | 1.35 |
| CPU time per trial (s) | $10^{-4}$ | $2 \times 10^{-4}$ |

← efficient !

(pay attention to the normalization of p(x)...)

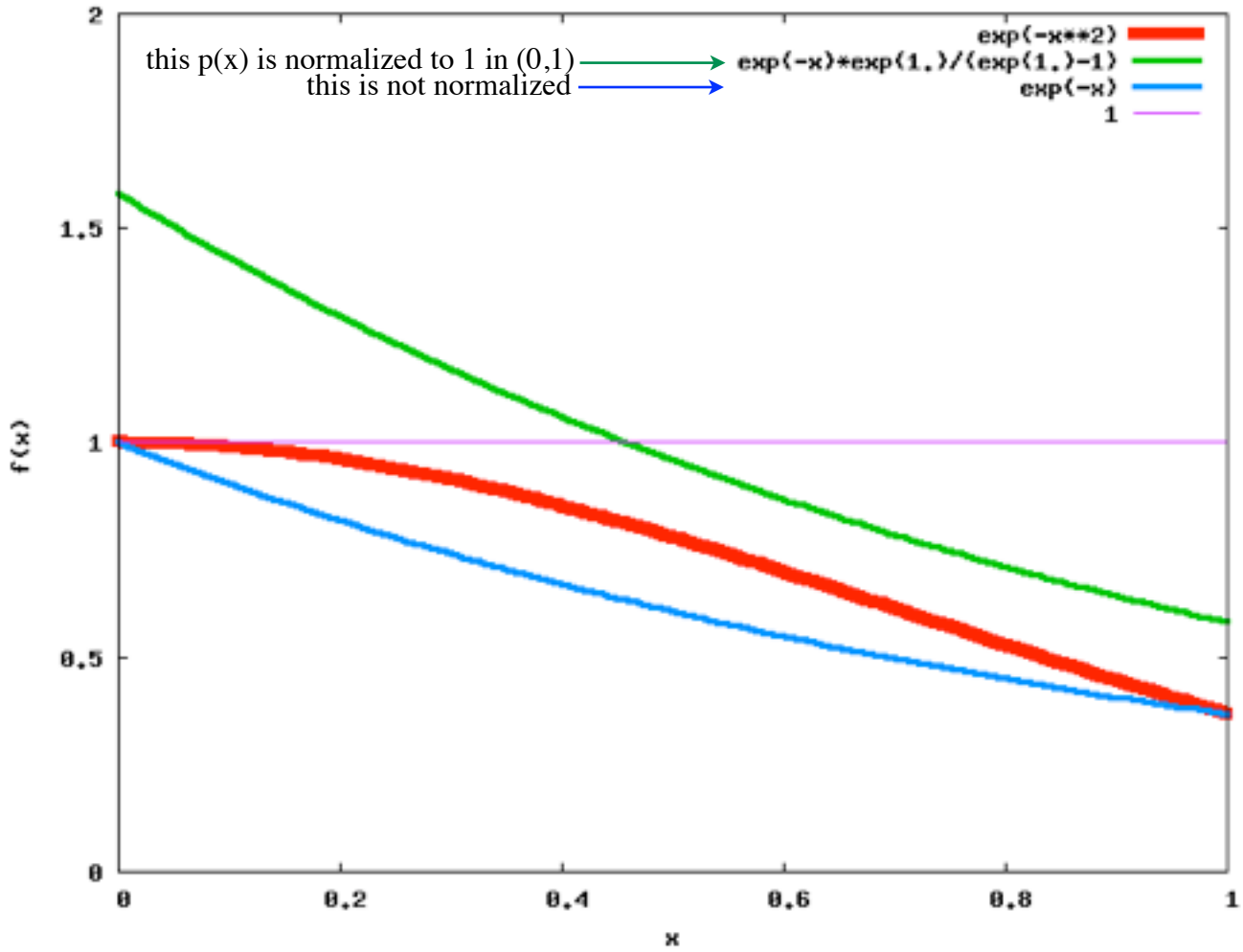
# error(MC)~1/$\sqrt{N}$ => see log(error) vs. log(N)
# but with different prefactors
# for sample means vs importance sampling



# Choice of the importance sampling function

$$F = \int_0^1 e^{-x^2}\, dx.$$

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




(pay attention to the normalization of p(x)...)

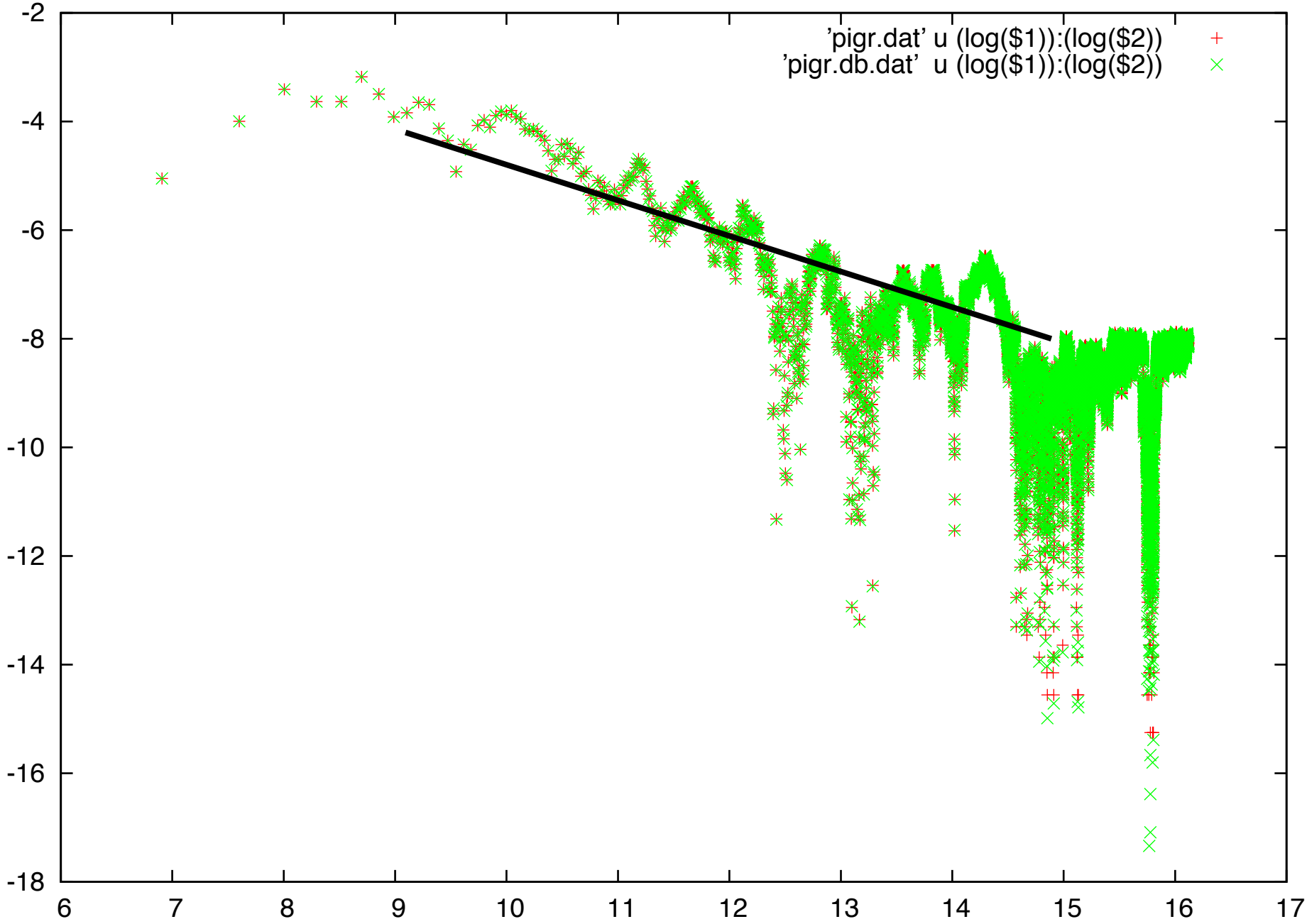
# Some programs

on https://moodle2.units.it/

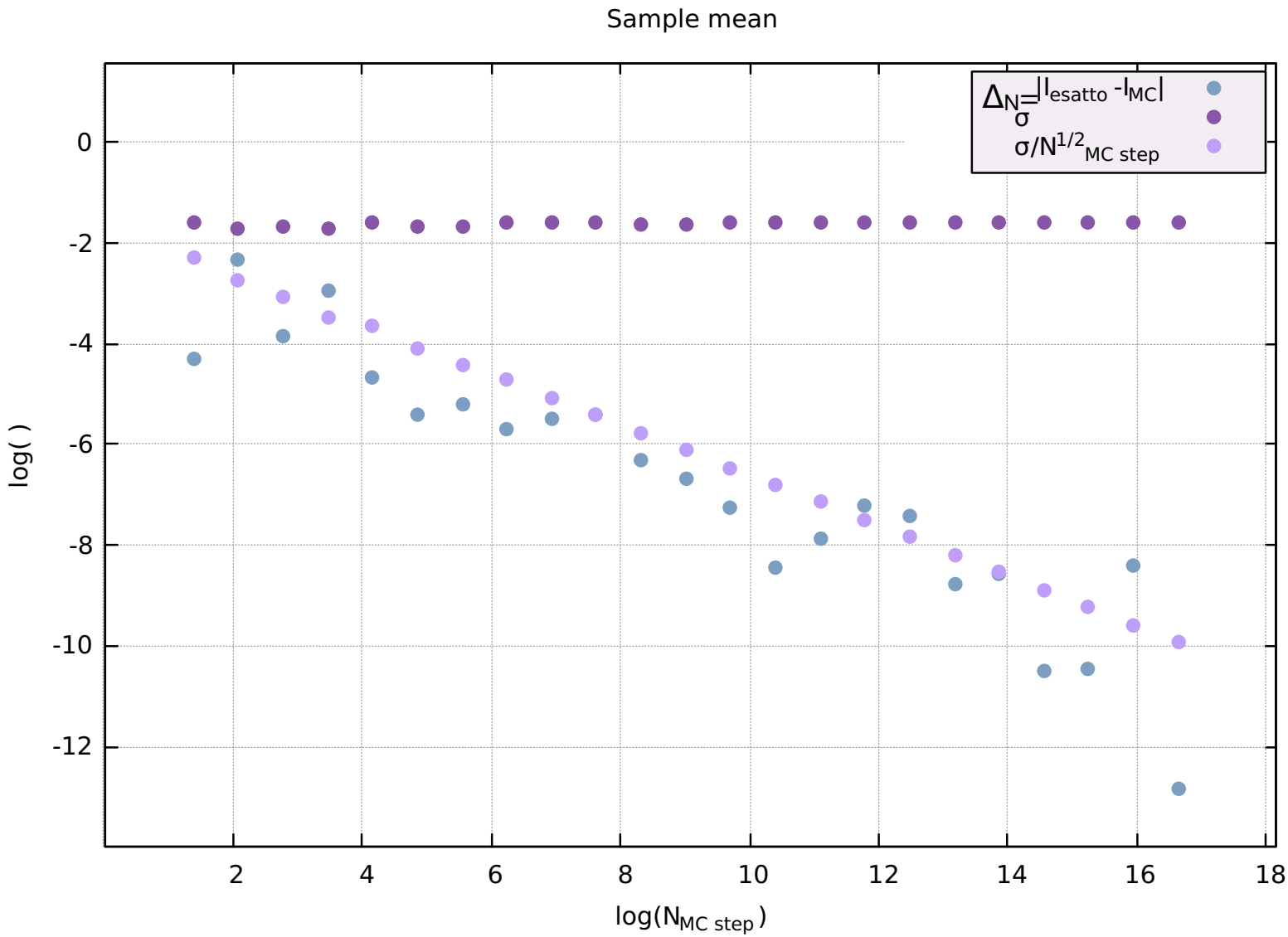
**pi.f90** Monte Carlo integration for the calculation of π

For the other exercises: write yourself the codes

# error(MC)~1/√N => see log(error) vs. log(N)



# error(MC)~1/√N :"true" error and statistical error




(credits: G. Lautizi, a.y. 2019-20)

# Summary of numerical integration (MC and deterministic) methods

## MC sample mean

$$\int_a^b f(x)dx = (b-a) < f > \approx (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i) \quad with\ \{x_i\}\ randomly\ uniformly\ distributed\ in\ [a,b]$$

$$\left(it\ can\ be\ considered\ as\ Importance\ sampling\ with\ p(x) = \frac{1}{b-a}\ in\ [a,b]\right)$$

## MC importance sampling

$$\int_a^b f(x)dx = \int_a^b \frac{f(x)}{p(x)}p(x)dx = < \frac{f(x)}{p(x)} > \int_a^b p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\int_a^b p(x)dx$$

$$with\ \{x_i\}\ \ randomly\ \ distributed\ \ according\ \ p(x)$$

## Deterministic, equispaced points

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} v_i f(x_i) \qquad with\ x_i = a + \frac{b-a}{N}i\ ,\ v_i\ to\ be\ determined$$

## Deterministic, non equispaced points

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} v_i f(x_i) \qquad with\ \{x_i\}\ ,\ \{v_i\}\ to\ be\ determined$$

# Error estimate: comparison between deterministic and MC methods in d-dimension

# Error estimate for numerical integration with deterministic methods

(Reminder from previous slides)

$$\int f(x)dx = F_n + error$$

How to evaluate the error? Consider the Taylor expansion of the integrand function and then integrate:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2}f''(x_i)(x - x_i)^2 + \dots ,$$

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = f(x_i)\Delta x + \frac{1}{2}f'(x_i)(\Delta x)^2 + \frac{1}{6}f''(x_i)(\Delta x)^3 + \dots \quad (*)$$

$$\Delta x \equiv x_{i+1} - x_i$$

# Error estimate for numerical integration: Rectangular approximation

(Reminder from previous slides)

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \boxed{f(x_i)\Delta x}$$

Compare ▭ with (*):

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = f(x_i)\Delta x + \frac{1}{2}f'(x_i)(\Delta x)^2 + \frac{1}{6}f''(x_i)(\Delta x)^3 + \dots$$

error

(leading order in $\Delta x$ )

For $n$ intervals $(\Delta x = (b-a)/n)$: error is $n(\Delta x)^2 \sim 1/n$

(...and similarly for higher-order approximations)

# Numerical integration:
## multidimensional integrals

$$F = \int_R f(x,y)\,dxdy$$

The rectangular approximation gives $\Delta x \Delta y \sim (\Delta x)^2 \sim 1/n$, being $n$ the number of parts (or pairs of points) of the integration domain:

$$\int_{x_i}^{x_{i+1}} \int_{y_i}^{y_{i+1}} f(x,y)dxdy \approx f(x_i,y_i)\Delta x \Delta y \qquad (*)$$

The Taylor expansion of the integrand function gives:

$$f(x,y) = f(x_i,y_i) + f'_x(x_i,y_i)(x-x_i) + f'_y(x_i,y_i)(y-y_i) + \dots$$

$$\int_{x_i}^{x_{i+1}} \int_{y_i}^{y_{i+1}} f(x,y)dxdy = f(x_i,y_i)\Delta x \Delta y + f'_x(x_i,y_i)\frac{(\Delta x)^2}{2}\Delta y + f'_y(x_i,y_i)\Delta x \frac{(\Delta y)^2}{2} + \dots \quad (**)$$

(*) against (**) => error
(leading order in $\Delta x$ )

For $n$ intervals: error is $n(\Delta x)^3 \sim 1/n^{1/2}$

# Numerical integration: multidimensional integrals

Therefore for rectangular approx.:

d=1: $error \sim 1/n$ d=2: $error \sim 1/n^{1/2}$

In general:

if the error decreases as $n^{-a}$ for $d = 1$, then the error decreases as $n^{-a/d}$ in $d$ dimensions.

**Classical formulas with equispaced points: slowly decreasing error for multidimensional integration !**

# Numerical integration: error in MC methods

$$\sigma_n/\sqrt{n} \approx \sigma_m \approx \sigma_s/\sqrt{s}$$

($\sigma_n$ is roughly constant with n ; for uncorrelated points, the variance of the averages goes like $\sim 1/n^{1/2}$ )

- The average function value

$$\langle f \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

- The average squared function value

$$\langle f^2 \rangle \equiv \frac{1}{N} \sum_{i=1}^{N} f^2(x_i)$$

- Estimate of the integrand (+/- standard error)

$$\int f \, dV \approx V \langle f \rangle \pm V \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}}$$

# Numerical integration:
## errors in multidimensional integrals

| $d$ | Rect. | Trap. | Simps. | MC |
|---|---|---|---|---|
| 1 | $1/n$ | $1/n^2$ | $1/n^4$ | $1/n^{1/2}$ |
| 2 | $1/n^{1/2}$ | $1/n$ | $1/n^2$ | $1/n^{1/2}$ |
| 4 | $1/n^{1/4}$ | $1/n^{1/2}$ | $1/n$ | $1/n^{1/2}$ |
| ... | ... | ... | ... | |

if the error decreases as $n^{-a}$ for $d = 1$, then the error decreases as $n^{-a/d}$ in $d$ dimensions.

the error for all Monte Carlo integration methods decreases as $n^{-1/2}$ independently of the integral.

**Monte Carlo convenient for multidimensional integration !**

# Summary: advantages of MC integration methods

- convergence as ~$N^{1/2}$ in any dimension regardless of the smoothness of the integrand
- simplicity: only two simple steps required (namely, producing a set of sampling points and evaluating the integrand function over such points)
- generality: sampling can be used even on domains that do not have a natural correspondence with the 'standard' domain $[0,1]^d$ and thus are not well-suited to numerical quadrature
- better suited than quadrature for integrands with singularities (importance sampling can handle this problem)
- flexibility: easy to add more points as needed (in the Gaussian quadrature, increasing the accuracy implies doing calculations from scratch)

# Metropolis Algorithm

by Metropolis, Rosenbluth, Rosenbluth, Teller and Teller (1953)

to generate random points with a given distribution

(motivation: related to the "importance sampling" integration method)

# Metropolis Algorithm

how to generate random points with a given distribution?

$$p(x)$$

**Idea: produce a random walk with points $\{x_i\}$ whose asymptotic probability distribution $p_N(x)$ of the occupied positions approaches $p(x)$ after a large number N of steps**

# Metropolis Algorithm

how to generate random points with a given distribution?

$$p(x)$$

**Idea: produce a random walk with points $\{x_i\}$ whose asymptotic probability distribution $p_N(x)$ of the occupied positions approaches $p(x)$ after a large number N of steps**

A **random walk** in general is defined by specifying a transition probability $T(x_i \rightarrow x_j)$ from one value $x_i$ to another value $x_j$ and the distribution of points $x_0, x_1, x_2, \ldots$ converges to a certain $p(x)$

Comment:

need to consider a RW more general than the 'standard' RW with length and probability fixed for each step.

Remind: a RW with fixed length and $p_{left} = p_{right}$ gives $P_N(x)$ that for large N tends to a gaussian distribution with a standard deviation that depends on N:

$$\sigma^2 = Dt; \quad D = \frac{\ell^2}{2\Delta t}; \quad \Delta t = \frac{t}{N} \Longrightarrow \sigma^2 = \ell^2 N/2 \qquad P(x, N\Delta t) = \sqrt{\frac{2}{\pi N}} e^{-x^2/(2N)}$$

(here $\ell$=1; remind also the factor of 2 due to discretization)

The recipe to obtain a gaussian distribution with given σ from simple RWs was to generate *several* RWs with the same N and do the histogram of their end-points.
The approach we are going to discuss now is something **different**, the focus being **one RW**.

# Markov chains

Consider a sequence of "configurations" $C=\{C_1, C_2, \ldots C_N\}$ stochastically generated, i.e. $C_{k+1}$ is obtained from the previous one, $C_k$, by making some random changes on the former.

The sequence is a **Markov chain** if the probability of making a transition from $C_k$ to $C_{k+1}$ is not dependent on how we arrived at $C_k$ (its history), i.e. no memory.

The sequence of points $x_0, x_1, x_2, \ldots$ of a simple RW is a Markov chain.

# The detailed balance

Choose a transition probability $T(x_i \rightarrow x_j)$ from one value $x_i$ to another value $x_j$ (from one configuration $C_i$ to another one $C_{i+1}$) such that the distribution of points $x_0, x_1, x_2, \ldots$ (of configurations) converges to the desired $p(x)$.

It is sufficient (not necessary) to satisfy the condition:

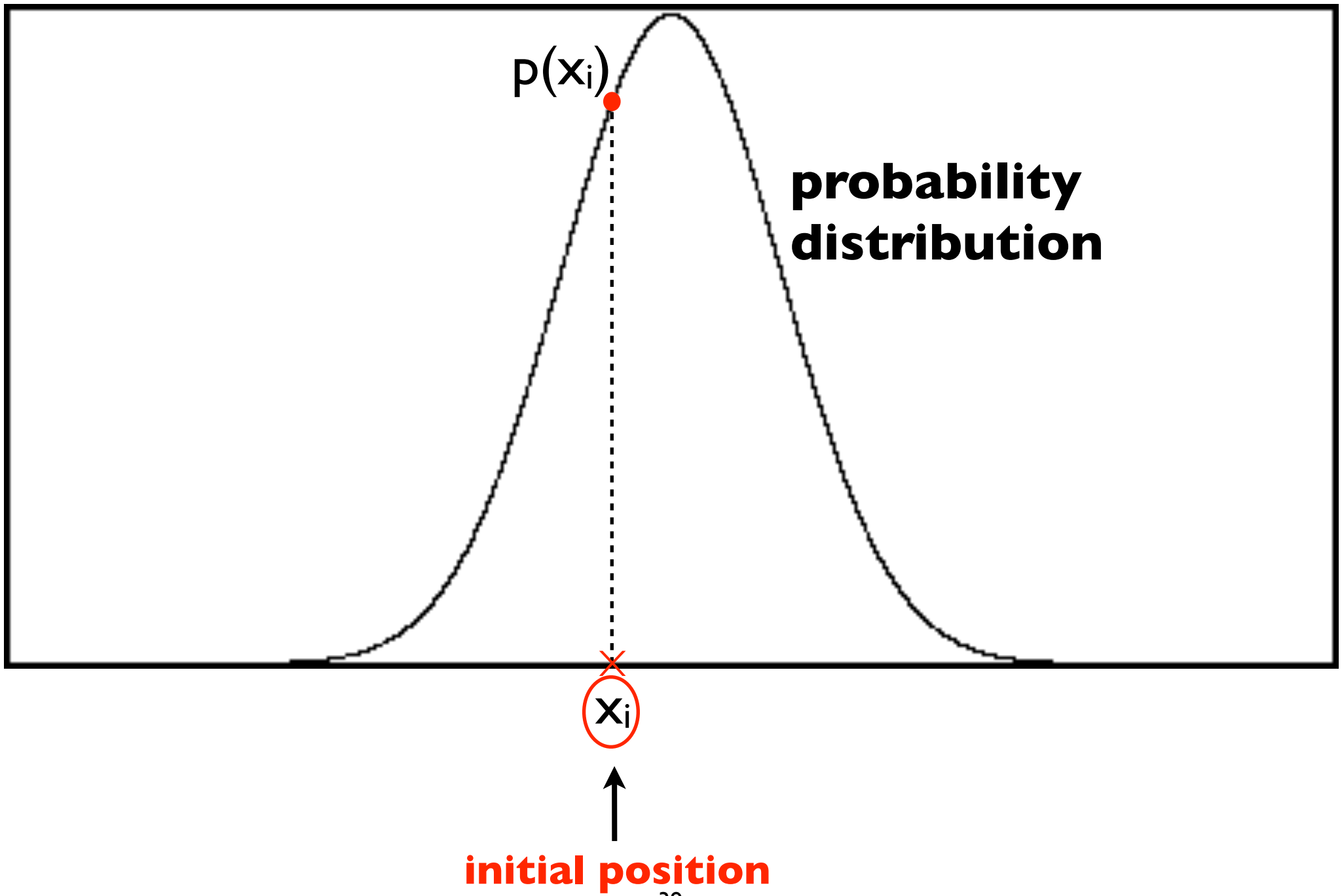
$$p(x_i)T(x_i \rightarrow x_j) = p(x_j)T(x_j \rightarrow x_i)$$

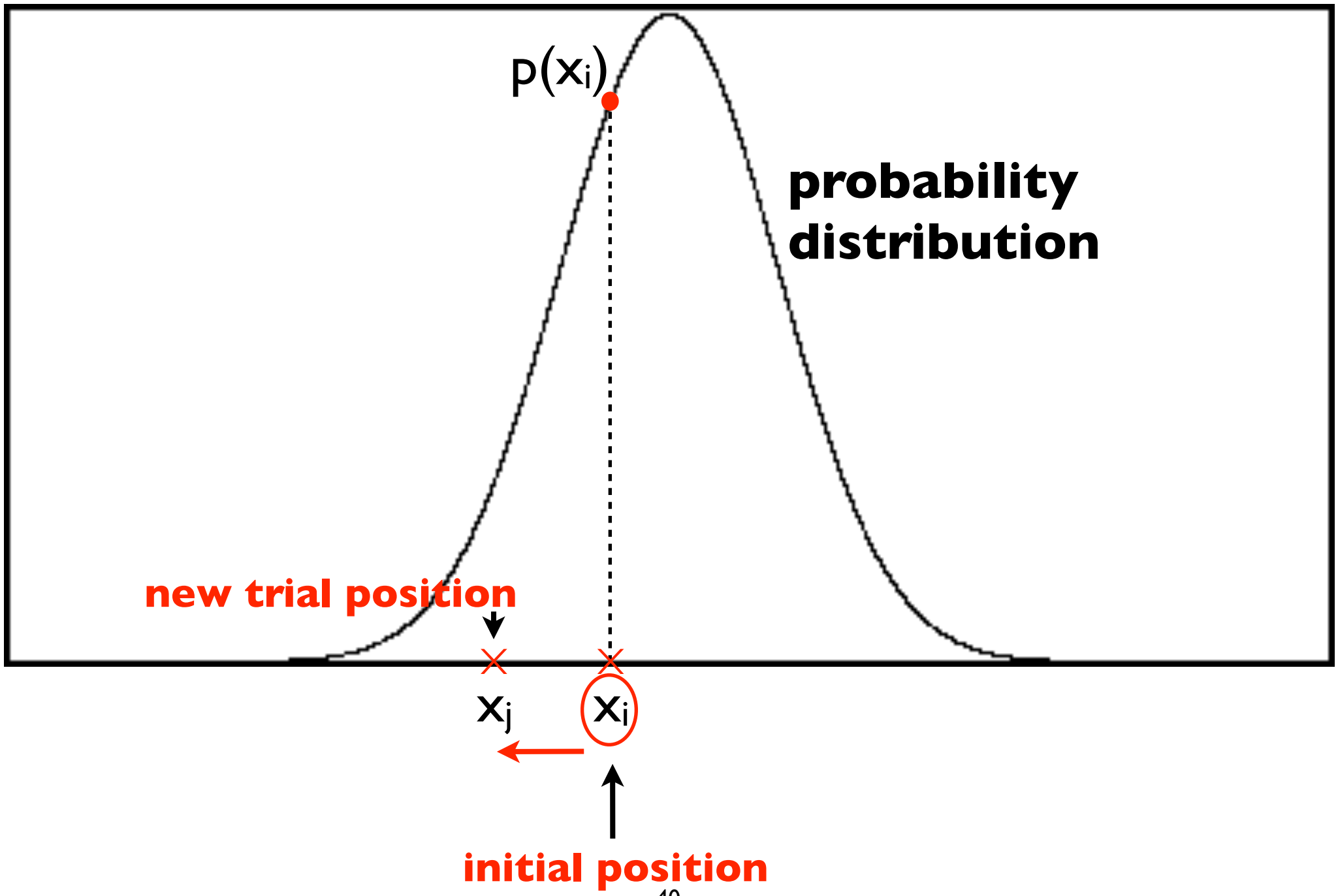
A simple choice (not unique!) is:

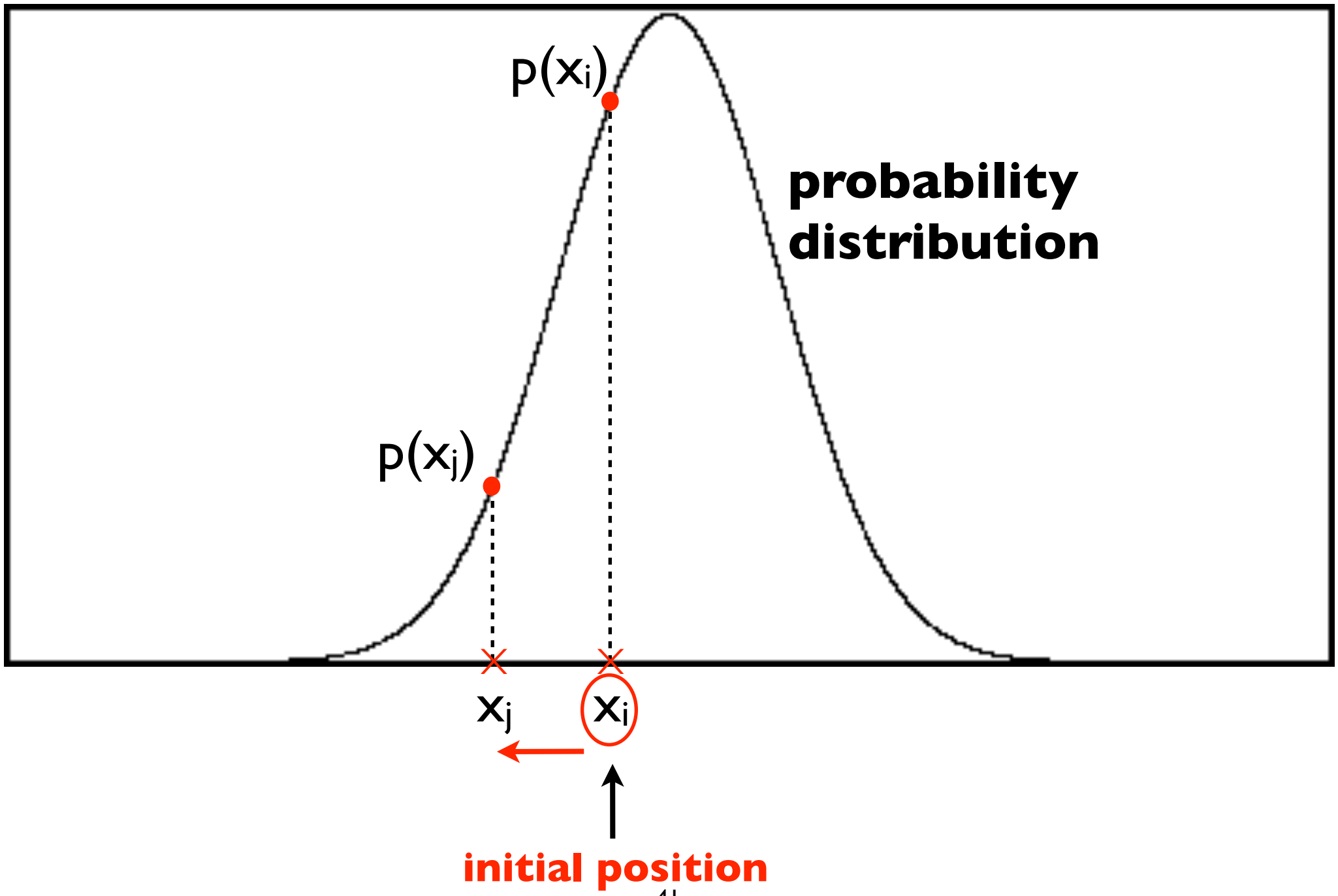
$$T(x_i \rightarrow x_j) = \min\left[1, \frac{p(x_j)}{p(x_i)}\right]$$

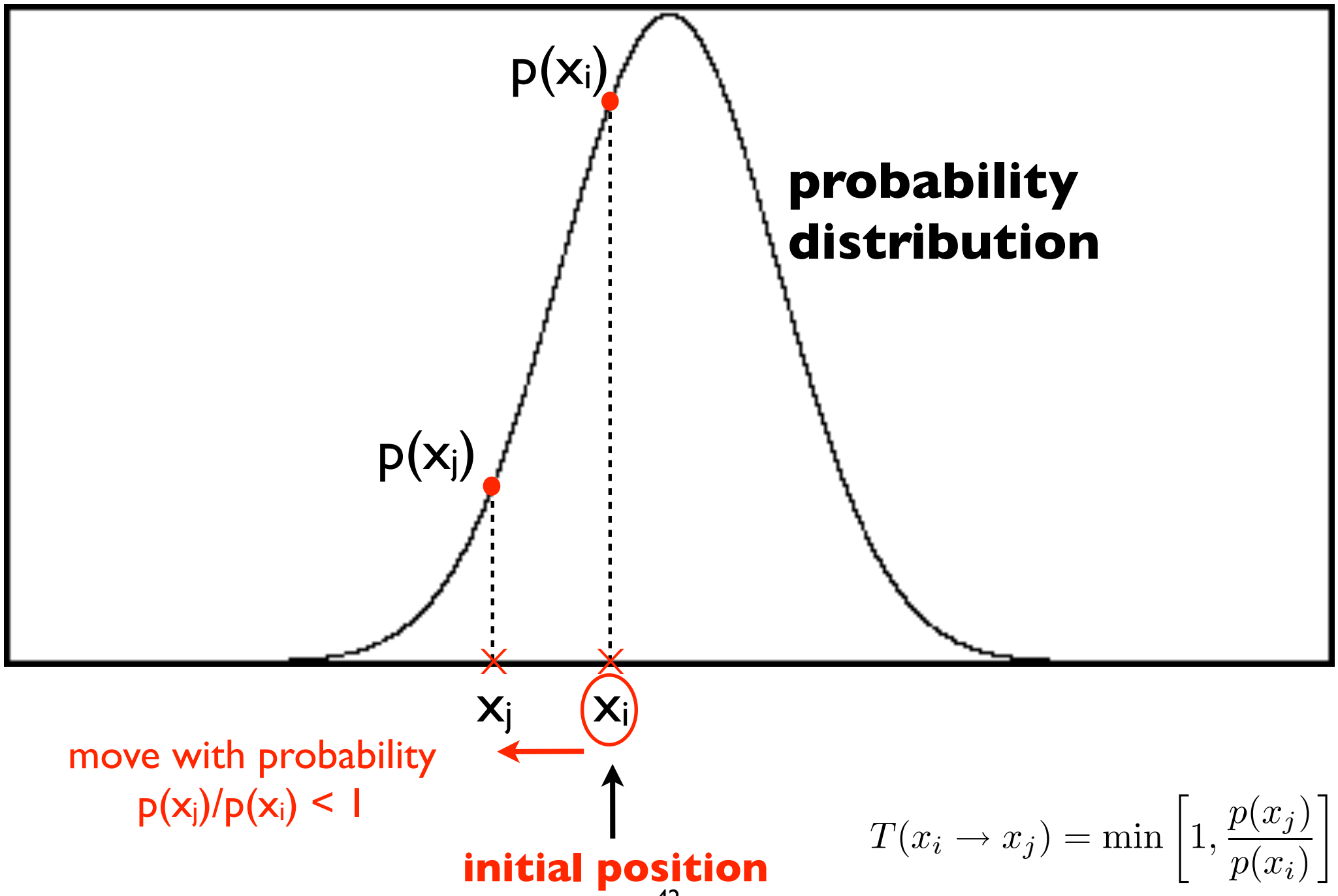
(We can easily verify...)

Example:



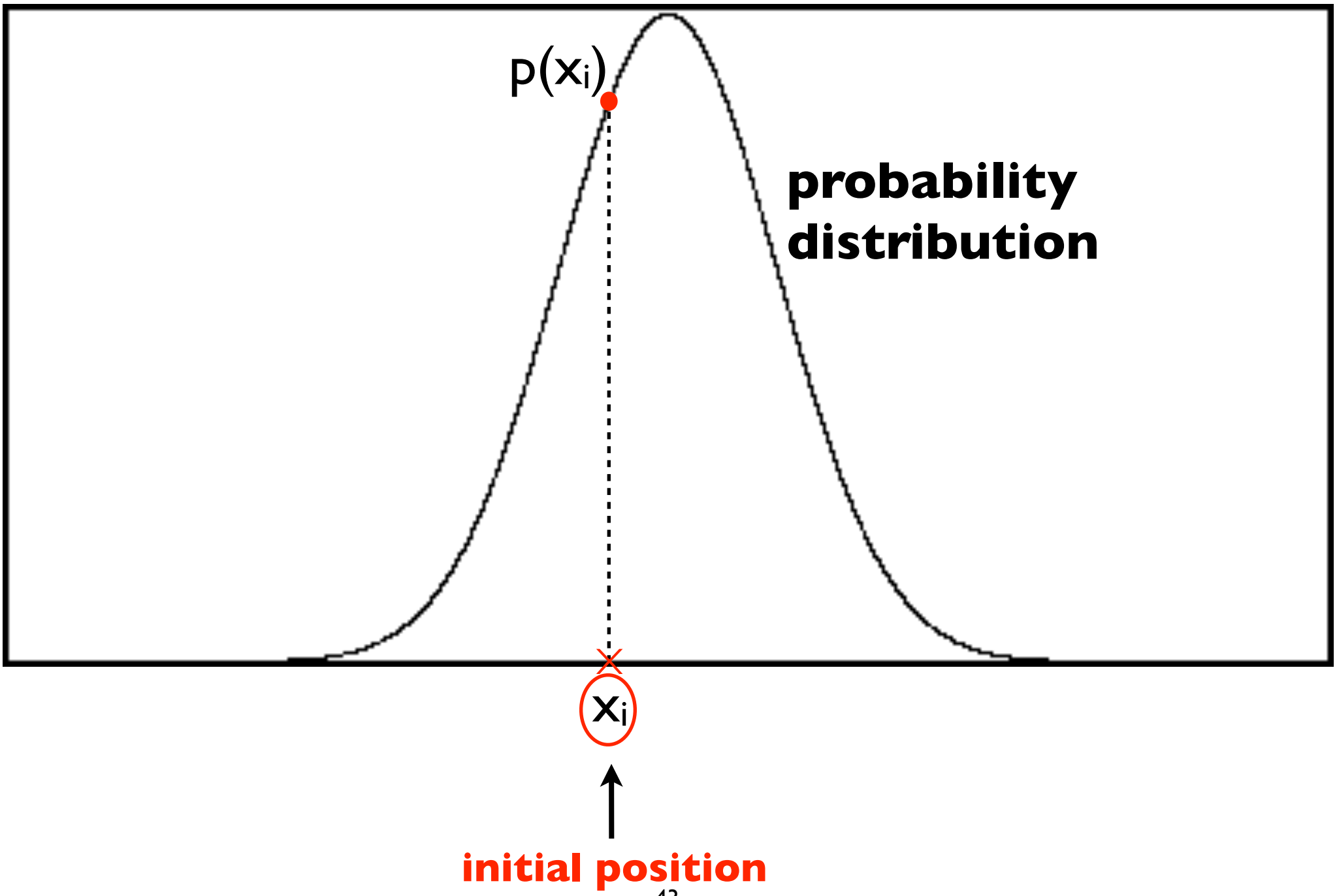

p(xi)
probability distribution
new trial position
xj
xi
initial position

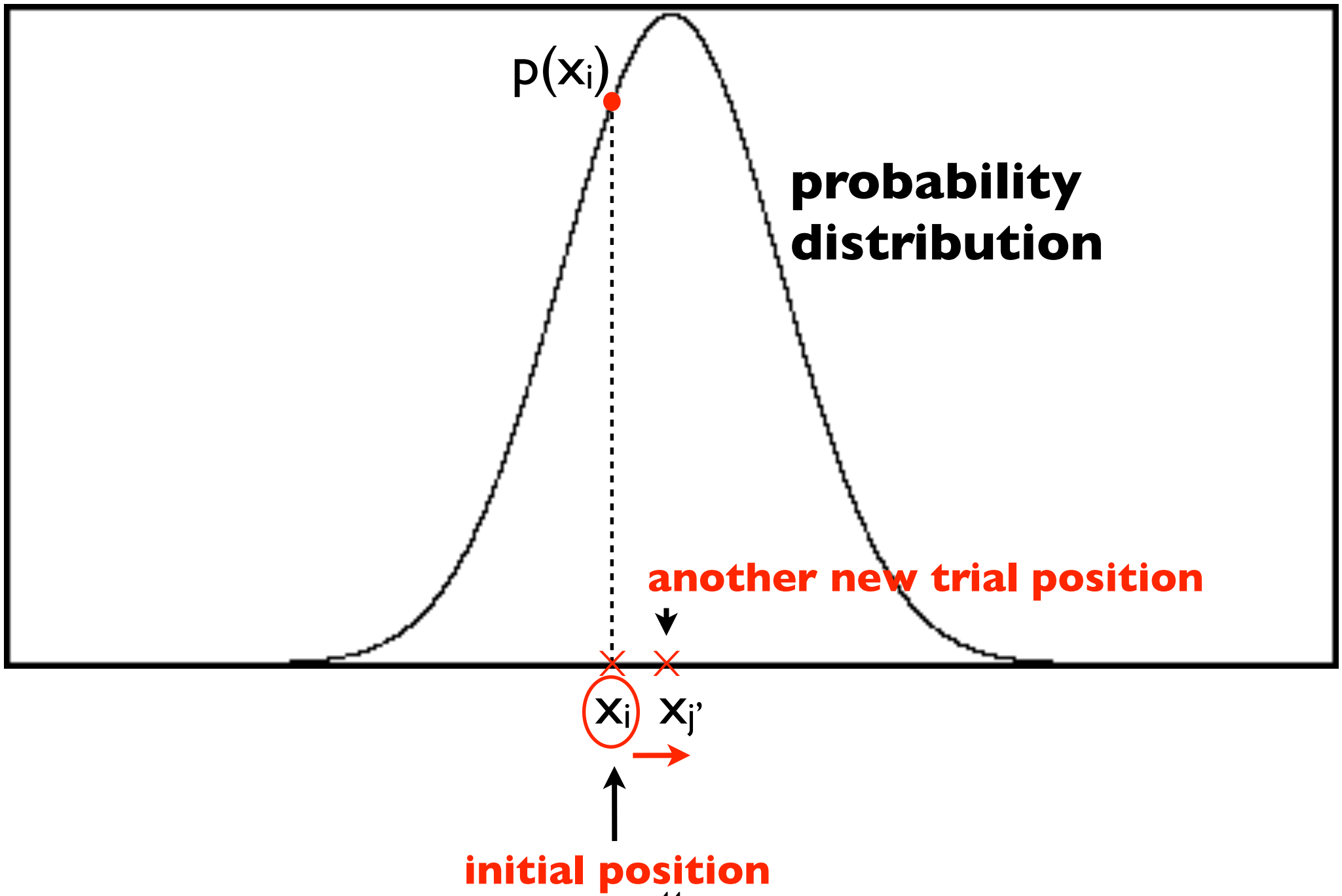

p(xi)
probability
distribution
p(xj)
xj
xi
initial position

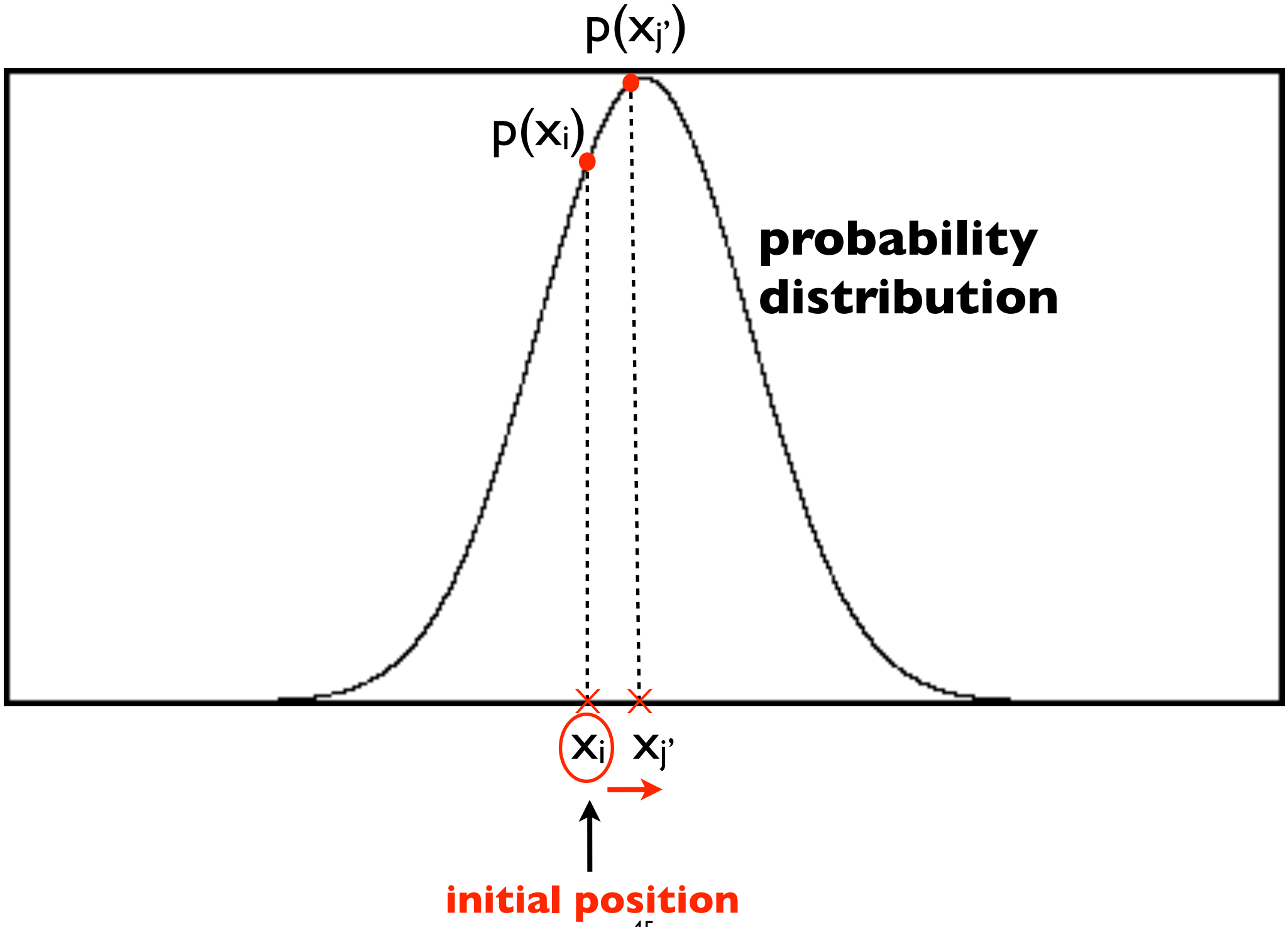


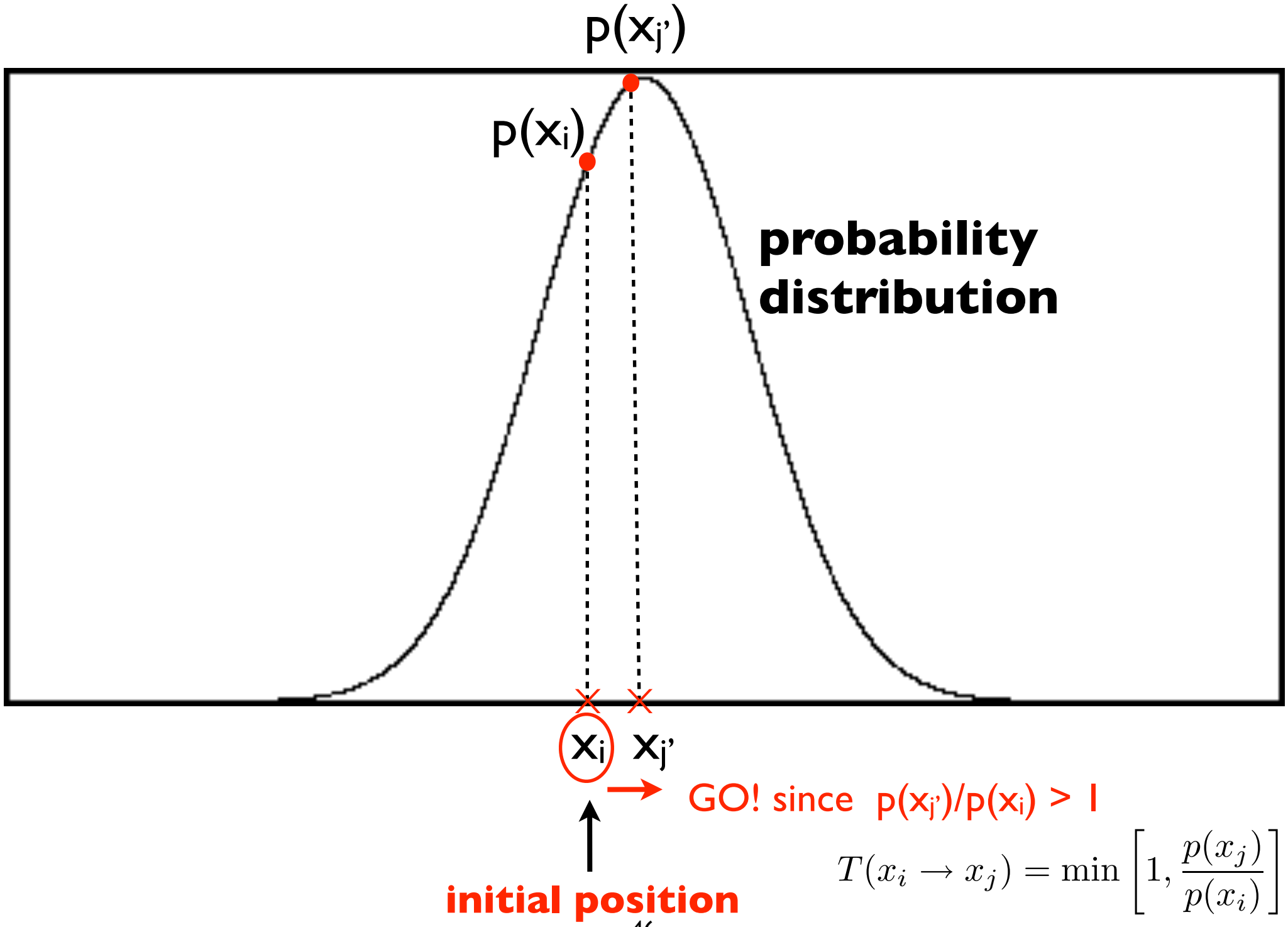

$$T(x_i \to x_j) = \min\left[1, \frac{p(x_j)}{p(x_i)}\right]$$

or:



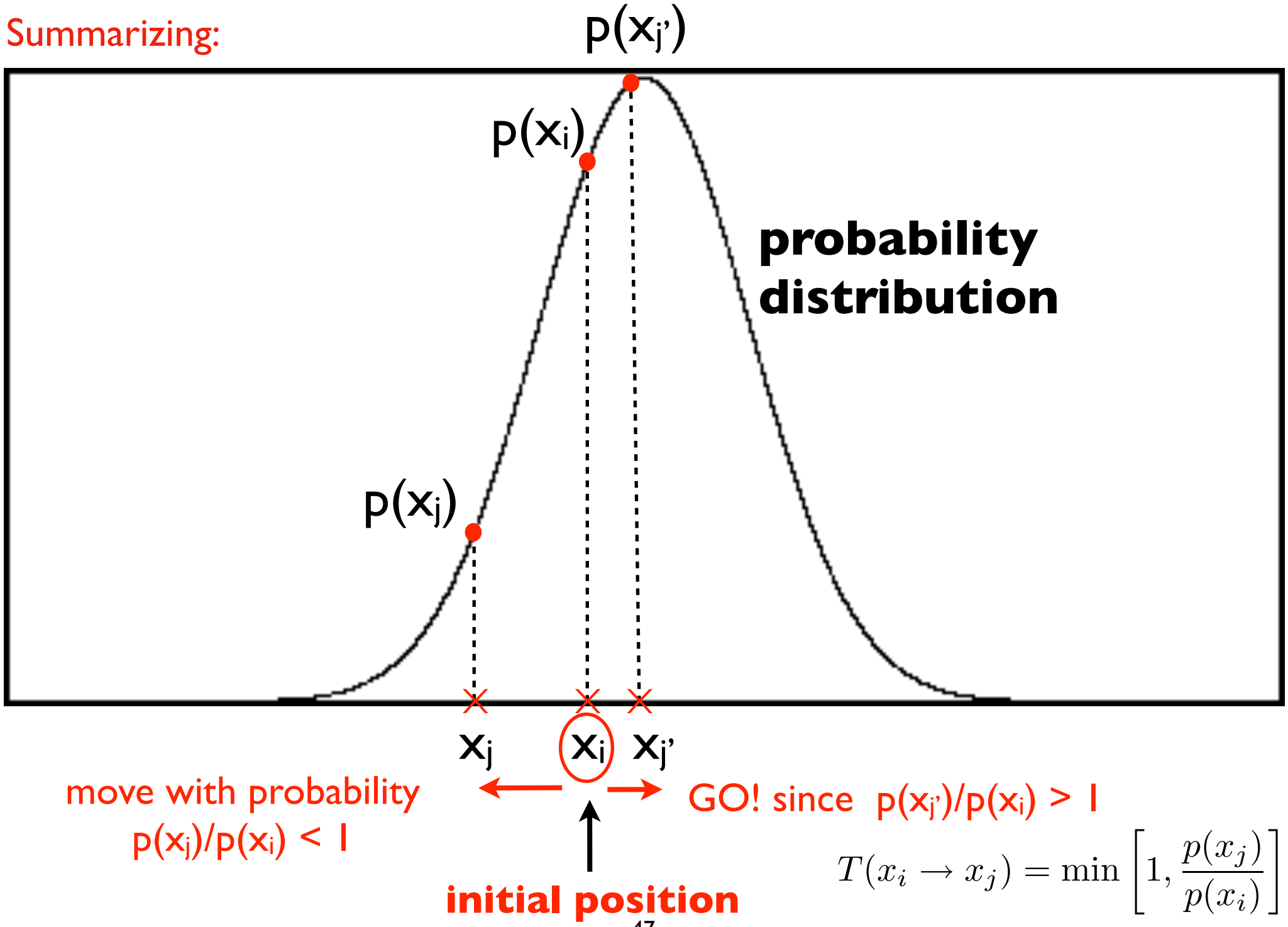

p(xi)
probability distribution
another new trial position
xi
xj'
initial position


p(xj')
p(xi)
probability distribution
xi
xj'
initial position




$$T(x_i \to x_j) = \min\left[1, \frac{p(x_j)}{p(x_i)}\right]$$

Summarizing:




$$T(x_i \rightarrow x_j) = \min\left[1, \frac{p(x_j)}{p(x_i)}\right]$$

# The Metropolis algorithm

$p(x)$ is given.

If the "walker" is at position $x_i$ and we wish to generate $x_{i+1}$, we can implement this choice of $T(x_i \rightarrow x_j)$ by the following steps:

1. Choose a trial position $x_{\text{trial}} = x_i + \delta_i$, where $\delta_i$ is a random number in the interval $[-\delta, \delta]$.
2. Calculate $w = p(x_{\text{trial}})/p(x_i)$.
3. If $w \geq 1$, accept the change and let $x_{i+1} = x_{\text{trial}}$.

else

4. If $w < 1$, generate a random number $r$.
5. If $r \leq w$, accept the change and let $x_{i+1} = x_{\text{trial}}$.
6. If the trial change is not accepted, then let $x_{i+1} = x_i$.

The algorithm from 1) to 6) has to be repeated until the distribution $p(x)$ of the points $\{x_i\}$ is reached.

# note:

it's important how to handle the rejected attempts for the generation of the random walk:

in case of a rejected attempt, the walker does not move, and we have to consider again the point where we tried to move from;

in the integration with importance sampling, a point which is unchanged after a rejected attempt does enter again in the average, i.e. its weight in the sum increases

# Questions:

- how to choose $x_0$ ?
- how to choose $\delta$ ?
  (if too small, most trial steps accepted, but the walker moves too slowly; if too large, only a few trial steps are accepted...)
- equilibration is necessary (how many steps?)

# Answers:

- how to choose $x_0$ ?
  Convenient to start from a maximum
- how to choose $\delta$ ?
  (if too small, most trial steps accepted, but the walker moves too slowly; if too large, only a few trial steps are accepted...)
  A good compromise is a choice accepting from ~ 1/3 to ~1/2 of the trial steps
- equilibration is necessary (how many steps?)
  A possible criterion based on error estimate

# Some programs:

in moodle2.units.it:

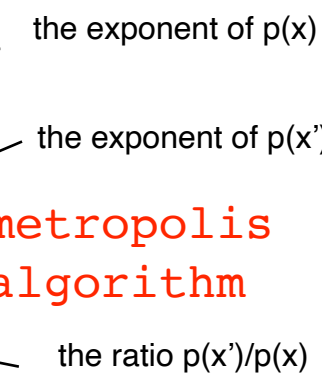
**gauss_metropolis.f90**

```
! gauss_metropolis.f90
! METROPOLIS generation of random numbers with a Gaussian distribution
! P(x) = exp(-x**2/(2*sigma**2))/sqrt(2*pi*sigma**2)
.... start from a given x=x0 .....
 do i=1,n

    !ccccccccccccccccccccccccccccc
    expx = - x**2 /(2*sigma**2)          !
    call random_number(rnd)              !
    xp = x + delta * (rnd-0.5)           !
    expxp = - xp**2 /(2*sigma**2)        !    metropolis
    w = exp (expxp-expx)                 !    algorithm
    call random_number(rnd)              !
    if (w > rnd) then                    !
       x = xp                            !
    !ccccccccccccccccccccccccccccc

    endif


  enddo
```



```
! gauss_metropolis.f90
! METROPOLIS generation of random numbers with a Gaussian distribution
! P(x) = exp(-x**2/(2*sigma**2))/sqrt(2*pi*sigma**2)
.... start from a given x=x0 .....
 do i=1,n
     x1 = x1 + x
     x2 = x2 + x**2
     x3 = x3 + x**3
     x4 = x4 + x**4
     !ccccccccccccccccccccccccccccc
     expx = - x**2 /(2*sigma**2)     !
     call random_number(rnd)         !
     xp = x + delta * (rnd-0.5)      !
     expxp = - xp**2 /(2*sigma**2)   !   metropolis
     w = exp (expxp-expx)            !   algorithm
     call random_number(rnd)         !
     if (w > rnd) then               !
        x = xp                       !
     !ccccccccccccccccccccccccccccc
        acc=acc+1.
     endif
     ibin = nint(x/deltaisto)
     if (abs(ibin) < maxbin/2) istog(ibin) = istog(ibin) + 1
  enddo
```

calculate some momenta

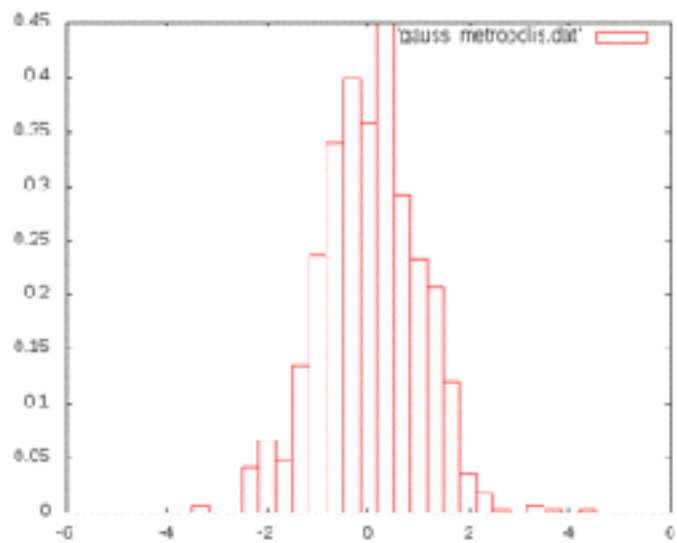
calculate the acceptance ratio

calculate the histogram

# Metropolis generation of random numbers distribution

1)

let's use the Metropolis method to generate a gaussian distribution




example of application:

(n=1000, x0=0, δ=5, σ=1)

(with **`gauss_metropolis.f90`**)

# Answers from numerical experiments:

- how to choose $x_0$?
  Convenient to start from a maximum
- how to choose $\delta$?
  (if too small, most trial steps accepted, but the walker moves too slowly; if too large, only a few trial steps are accepted...)
  A good compromise is a choice accepting from ~ 1/3 to ~1/2 of the trial steps: depends on $\sigma$
- equilibration is necessary (how many steps?)
  A possible criterion based on error estimate:
  consider when $\langle x^2 \rangle \approx \sigma^2$