993SM - Laboratory of Computational Physics lecture 7 - part 1 April 22, 2020

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

Metropolis algorithm

1) to generate random points with a given distribution

2) to calculate averages with importance sampling

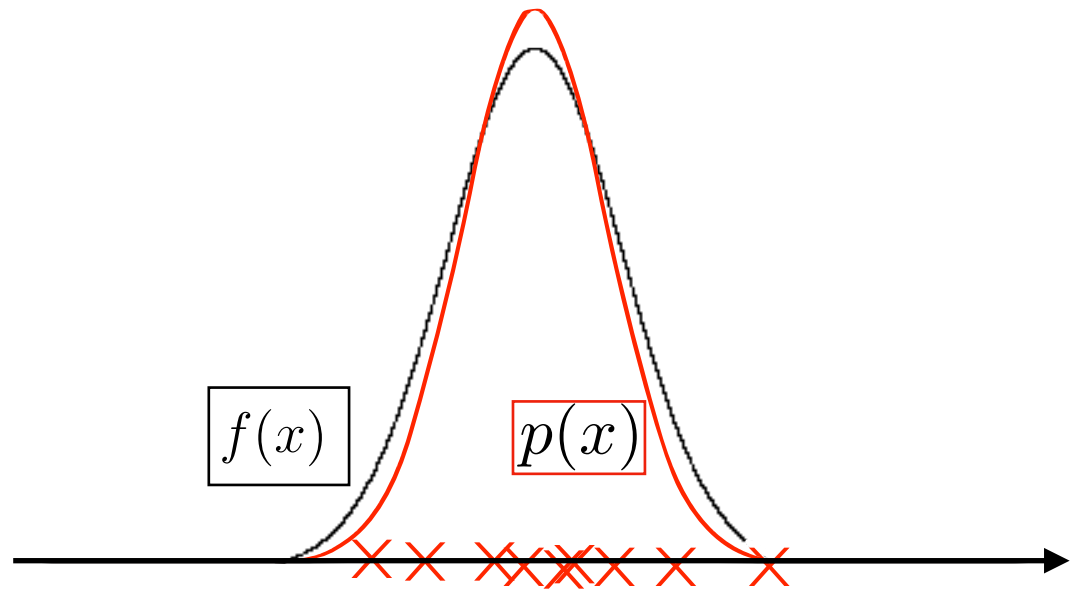
3) in particular: in the canonical ensemble

Metropolis Algorithm

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

A special case of importance sampling where certain possible sampling attempts are rejected.

- generate points according $p(x)$
- importance sampling of $f(x)$ using $p(x)$



Metropolis algorithm

1) to generate random points with a given distribution

2) to calculate averages with importance sampling

3) in particular: in the canonical ensemble

Metropolis Algorithm

1) 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

1) 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 ℓ=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 $p(x)$.

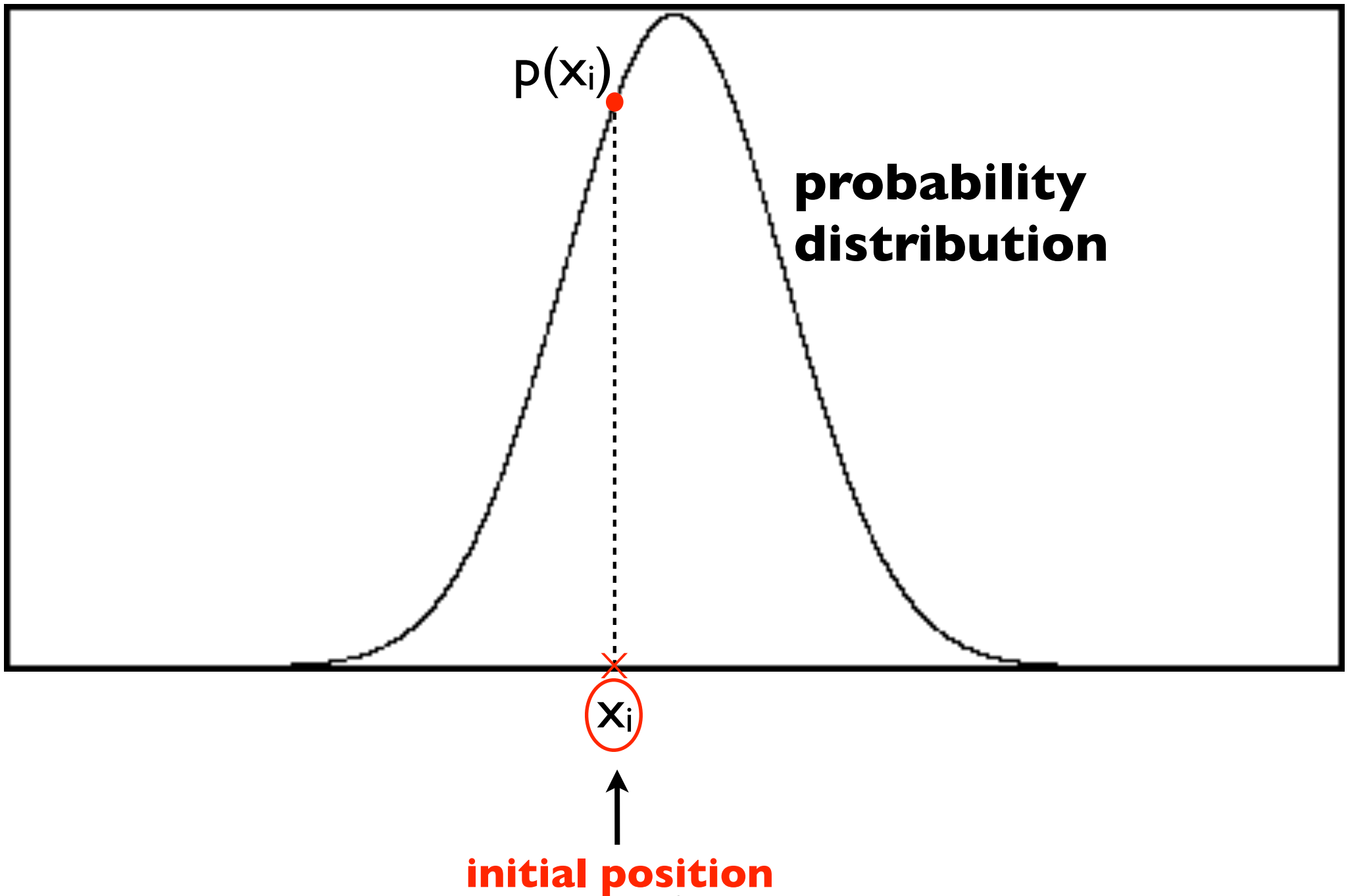
It is sufficient (not necessary) to satisfy the condition:

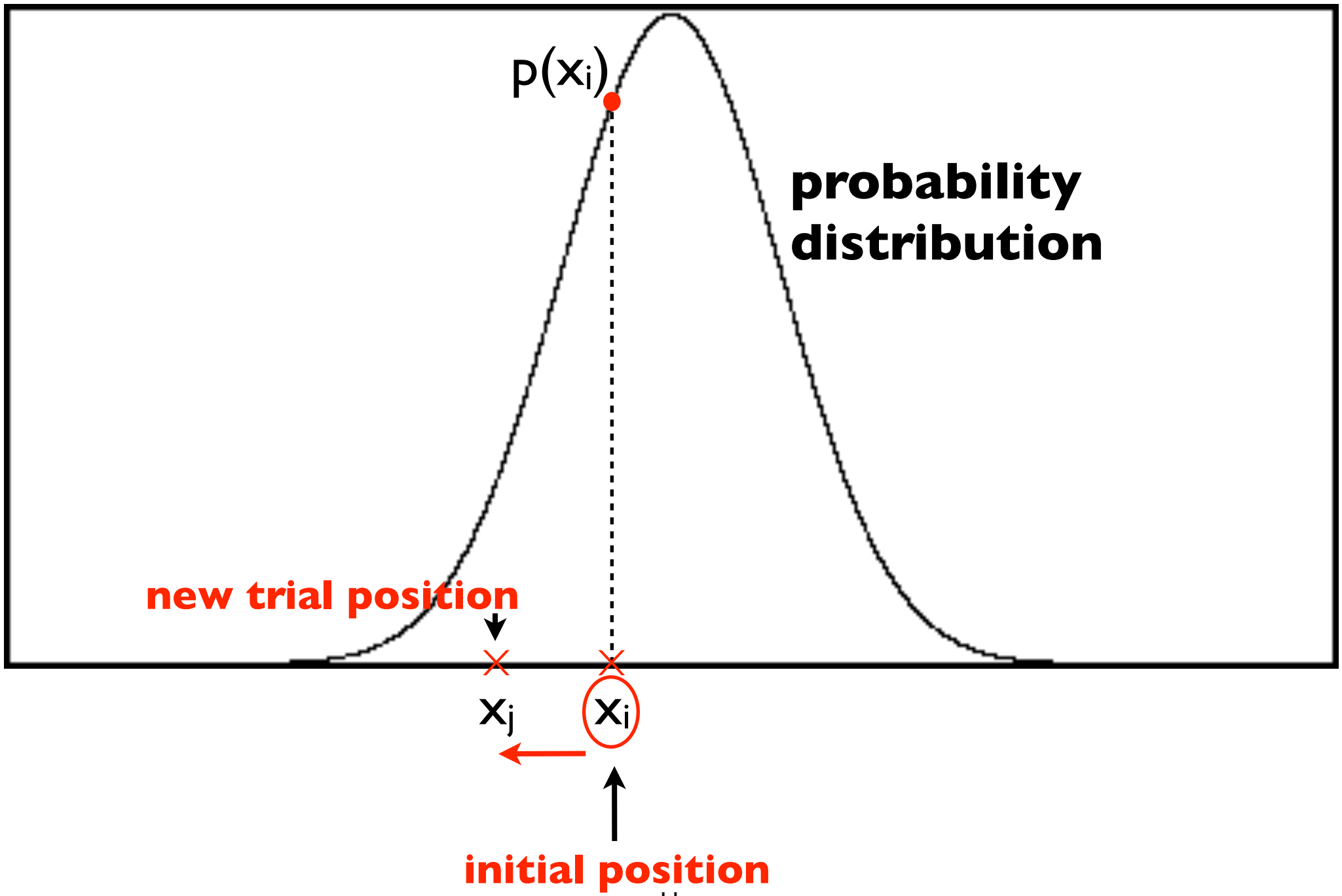
$$p(x_i)T(x_i \rightarrow x_j) = p(x_j)T(x_j \rightarrow x_i)$$

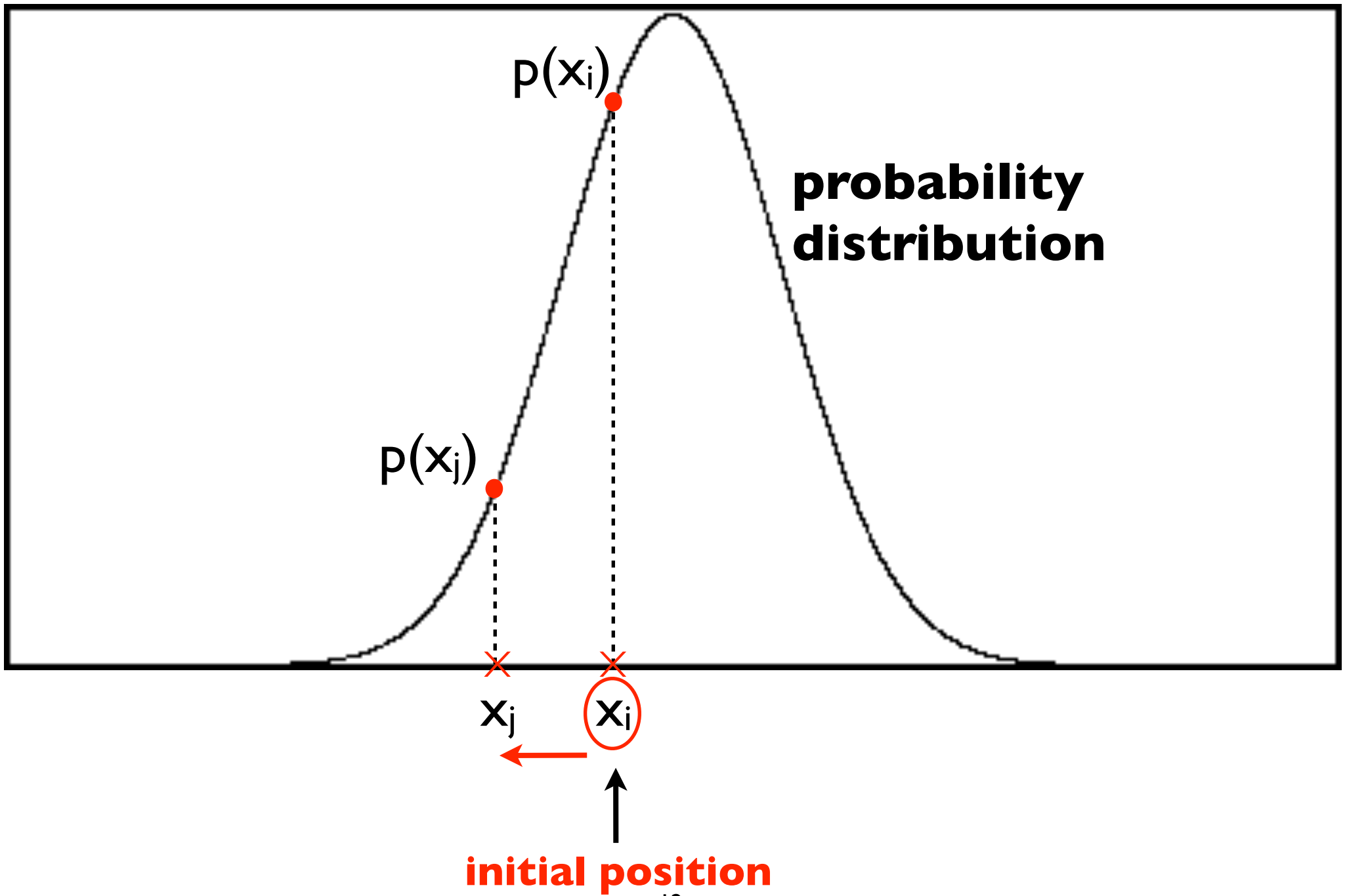
A simple choice (not unique!) is:

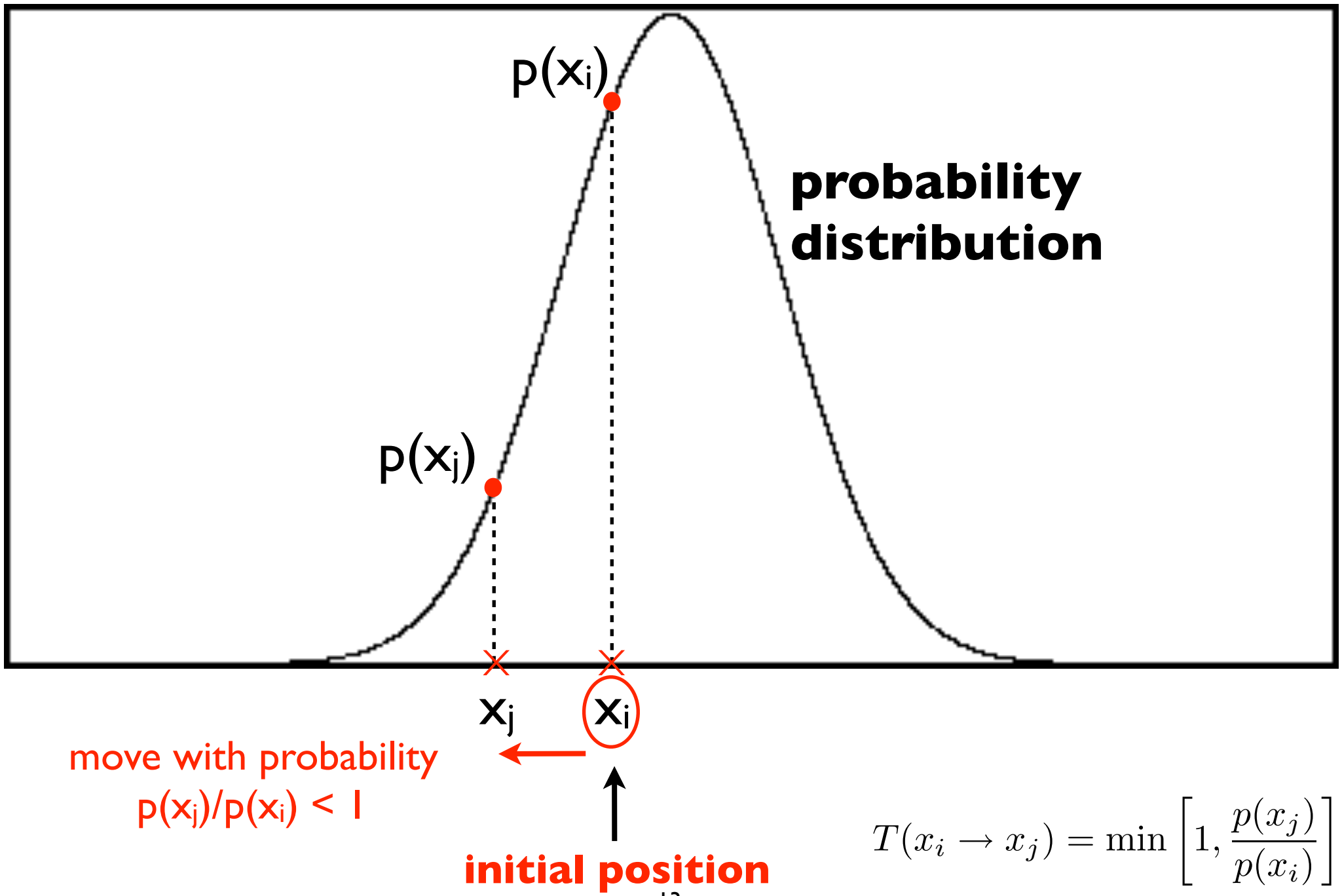
$$T(x_i \rightarrow x_j) = \min\left[1, \frac{p(x_j)}{p(x_i)}\right]$$

(We can easily verify...)

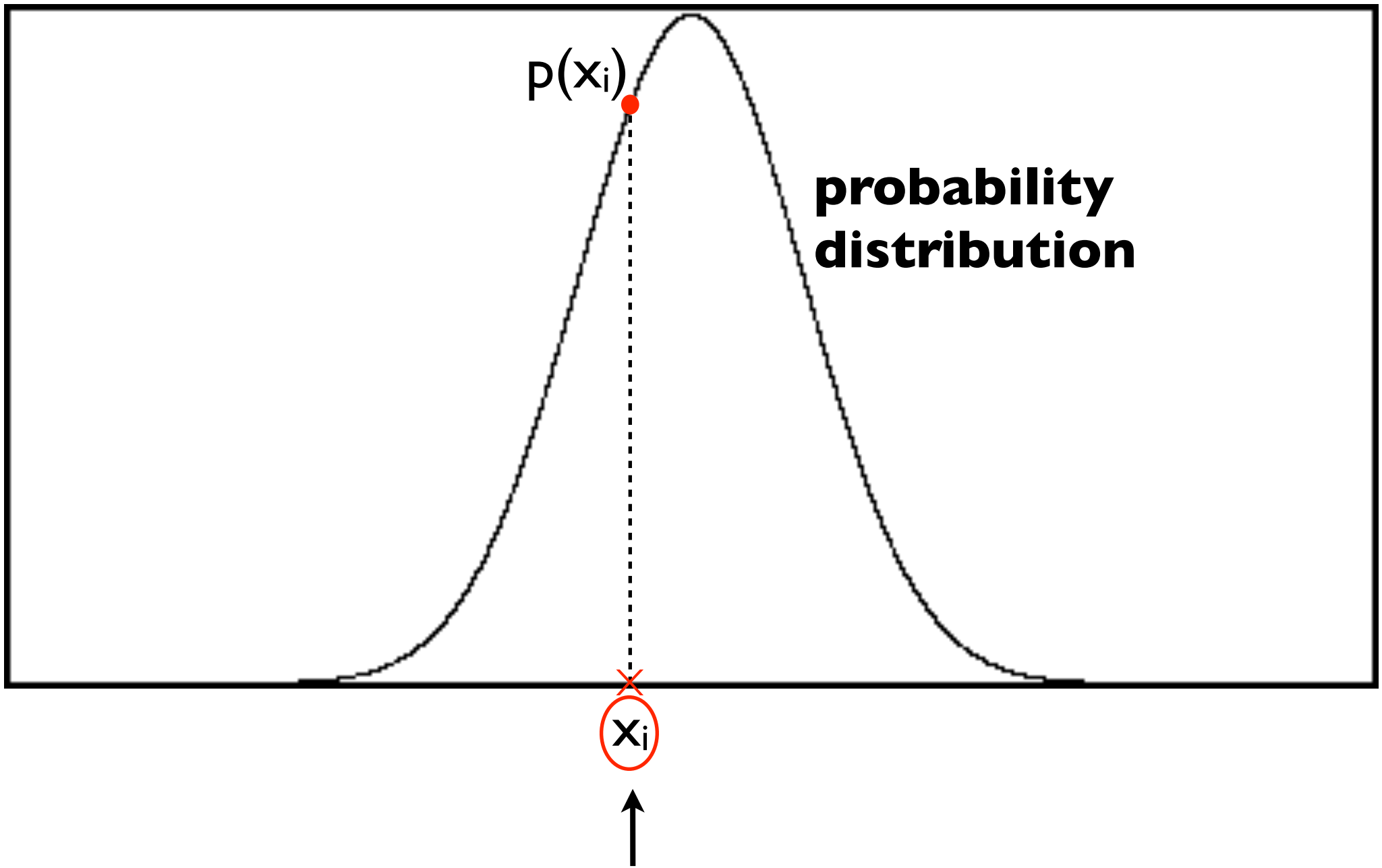

p(xi)
probability
distribution
xi
initial position

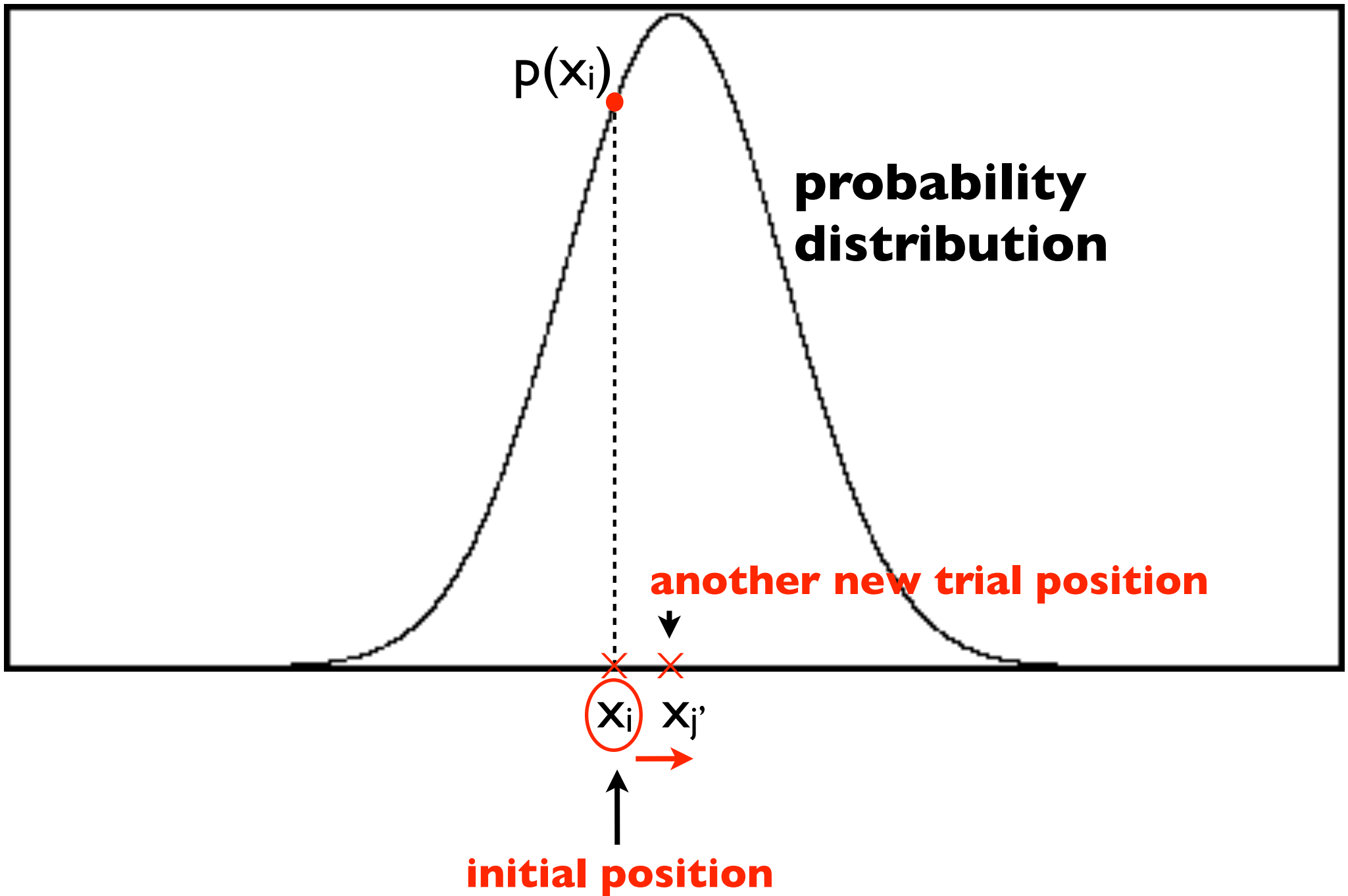

p(xi)
probability
distribution
new trial position
xj
xi
initial position

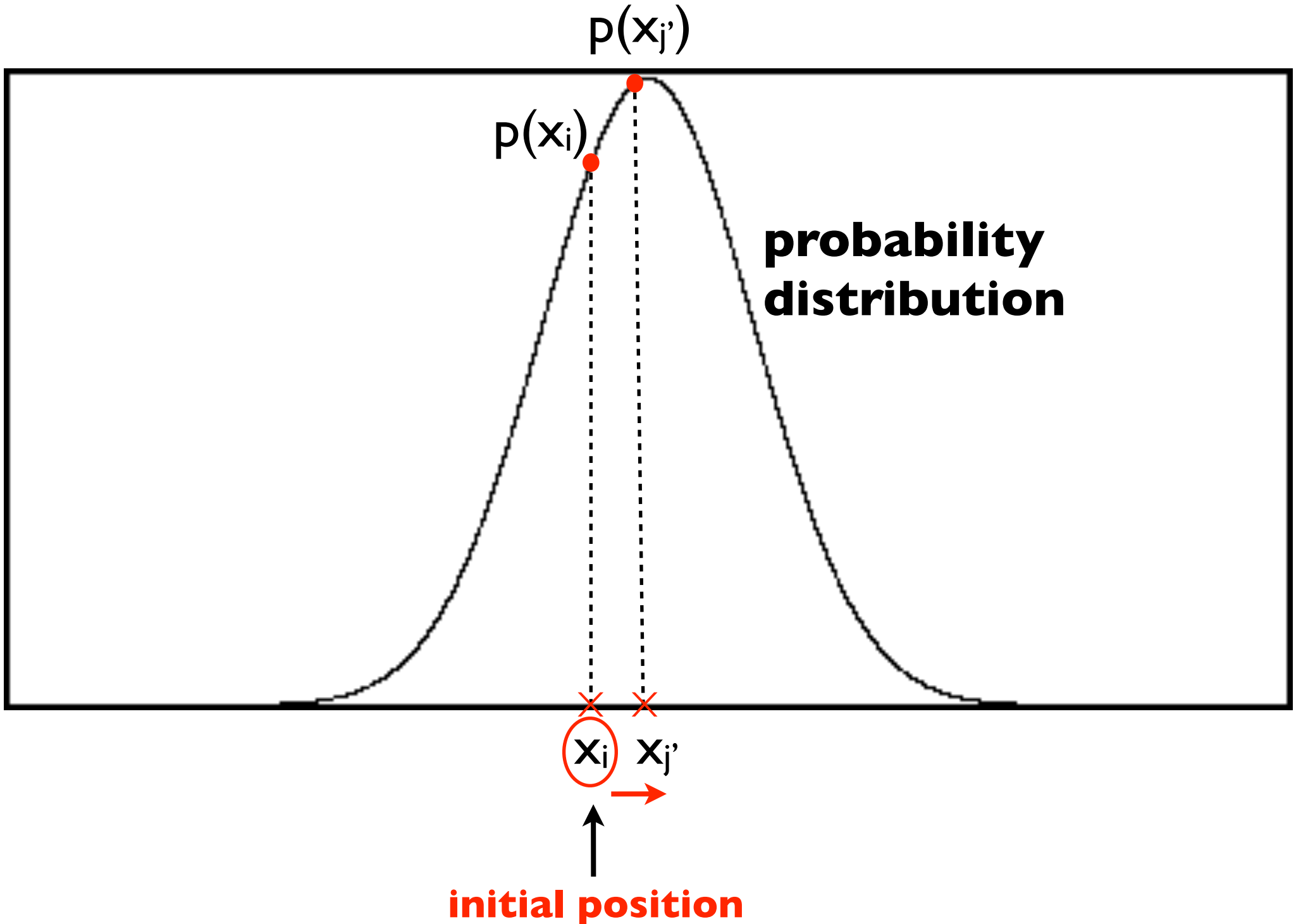

p(xi)
probability
distribution
p(xj)
xj
xi
initial position

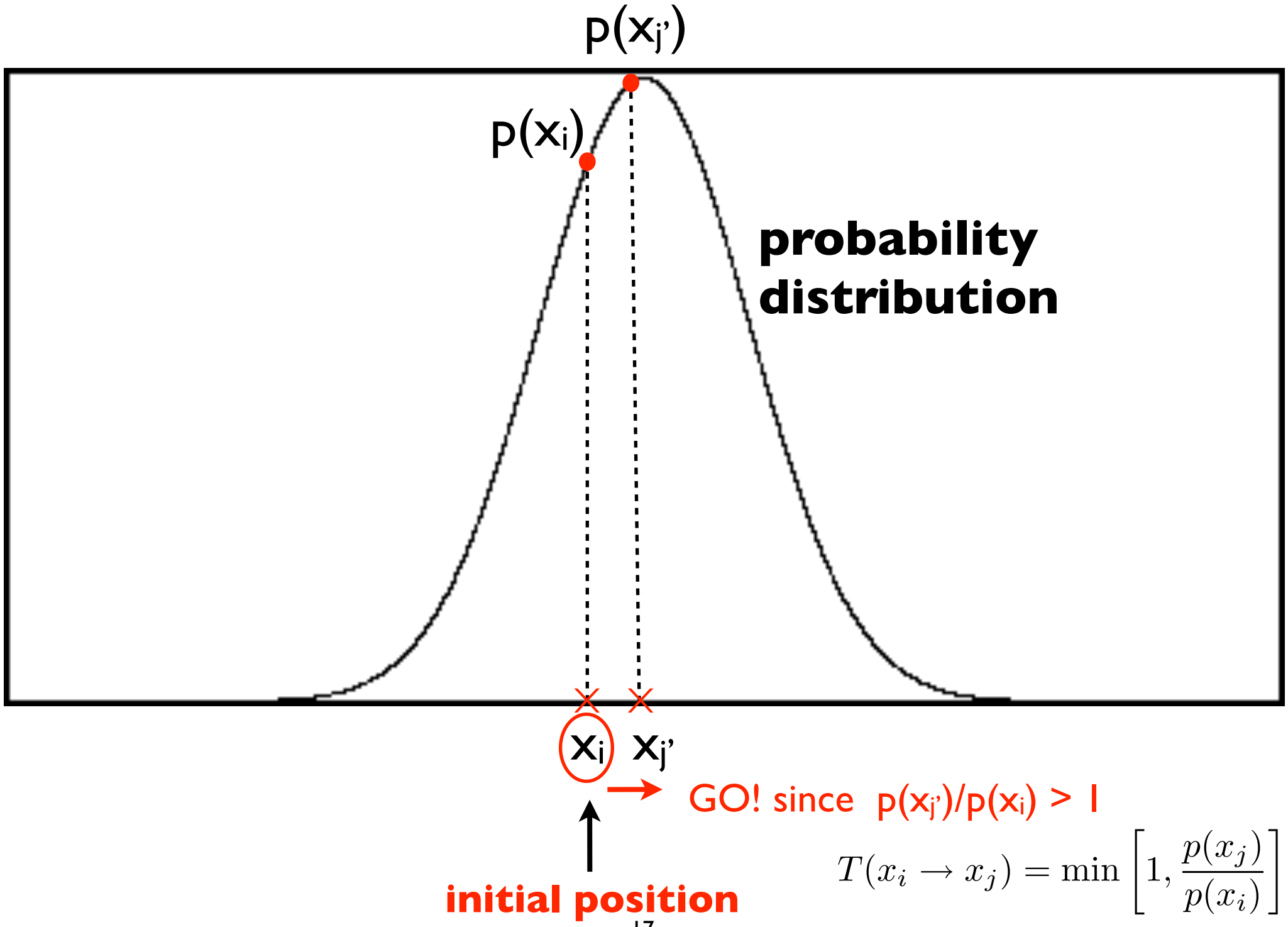


$$T(x_i \to x_j) = \min\left[1, \frac{p(x_j)}{p(x_i)}\right]$$

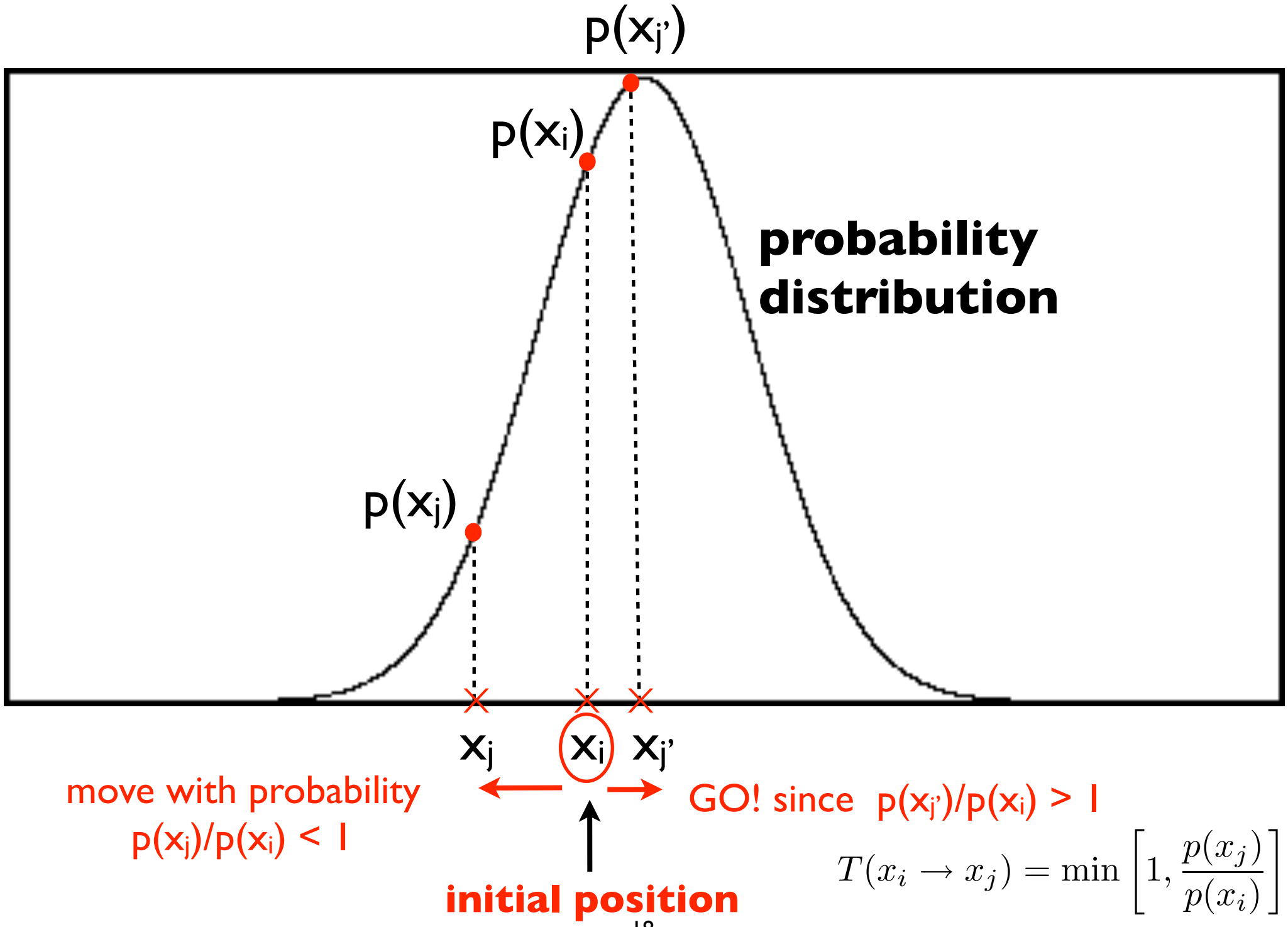

p(xi)
probability
distribution
xi
initial position


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


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



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



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

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 δ_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 δ ?
  (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 δ ?
  (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:

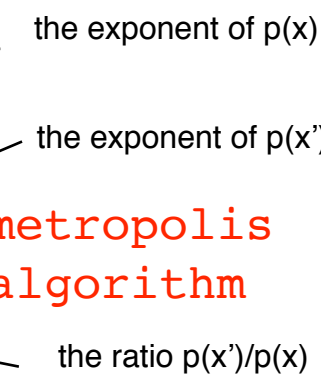
on
$/home/peressi/comp-phys/VII-metropolis/
[do: $cp /home/peressi/.../VII-metropolis/* .]
or in moodle2.units.it

gauss_metropolis.f90

metropolis_sampling.f90
direct_sampling.f90
boltzmann_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

    !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

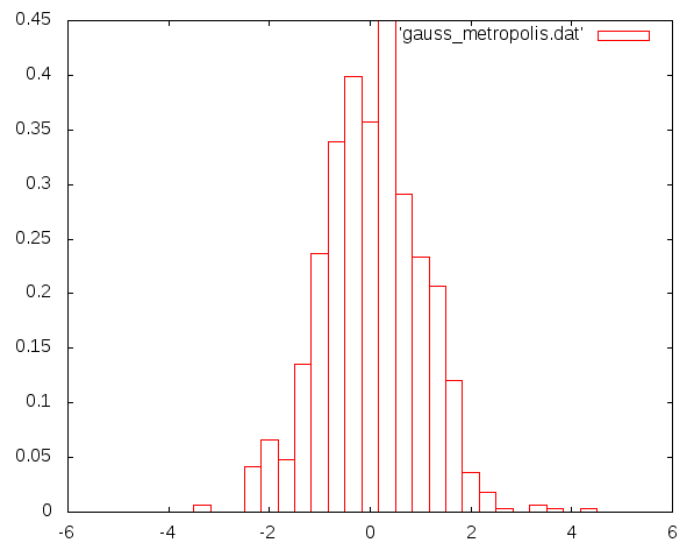
calculate the acceptance ratio

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 δ?
  (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 σ
- equilibration is necessary (how many steps?)
  A possible criterion based on error estimate:
  consider when $\langle x^2 \rangle \approx \sigma^2$

Metropolis algorithm

1) to generate random points with a given distribution

2) to calculate averages with importance sampling

3) in particular: in the canonical ensemble

example of application of the Metropolis algorithm

1) to generate arbitrary probability distributions

2) to compute averages

of the form $\langle f \rangle = \dfrac{\int p(x) f(x)\, dx}{\int p(x)\, dx}$

where the probability distribution $p(x)$ does not need to be normalized

(here: 1D, but generally useful also for multidimensional integrals)

reminder from Lecture V:

"importance sampling"

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

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

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

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

Here : substituting $\frac{f(x)}{p(x)} \Longrightarrow f(x)$ and rewriting, we have : $\langle f \rangle = \frac{\int p(x) f(x)\, dx}{\int p(x)\, dx}$

Some programs:

on
$/home/peressi/comp-phys/VII-metropolis/
[do: $cp /home/peressi/.../VII-metropolis/* .]
or in moodle2.units.it

gauss_metropolis.f90

metropolis_sampling.f90
direct_sampling.f90
boltzmann_metropolis.f90

Metropolis Sampling

Using a method to generate a distribution $p(x)$, we can efficiently sample integrals of the form

$$\langle f \rangle = \frac{\int p(x) f(x)\, dx}{\int p(x)\, dx}$$

example of application:

See **`metropolis_sampling.f90`** in the exercises:

example of estimate of average position, average kinetic, potential and total energies in the ground state of the harmonic oscillator:

$f(x)$: physical quantity; $p(x) = |\psi(x)|^2$

Metropolis Sampling

$$\langle f \rangle = \frac{\int p(x) f(x)\, dx}{\int p(x)\, dx}$$

$f(x)$: physical quantity; $p(x) = |\psi(x)|^2$

Consider the hamiltonian : $\mathcal{H} = -\frac{1}{2}\nabla^2 + \frac{1}{2}x^2$

σ here is a given input parameter

Consider a wavefunction (not necessarily the ground state) : $\psi(x) = \exp(-x^2/4\sigma^2)$

4 is ok

Interesting physical quantities are related to E_{pot}, E_{kin} , E_{tot}

The potential energy can be considered as f(x) (it is a multiplicative operator); kinetic and total energy can not, but their expectation value can be related to the average value of x^2:

$$\langle E_{pot} \rangle = \frac{\langle \psi | \frac{1}{2}x^2 | \psi \rangle}{\langle \psi | \psi \rangle} = \frac{\int \frac{1}{2}x^2 |\psi(x)|^2 dx}{\int |\psi(x)|^2 dx}$$

$$\langle E_{kin} \rangle = \frac{\langle \psi | -\frac{1}{2}\nabla^2 | \psi \rangle}{\langle \psi | \psi \rangle} = \frac{\int \left(\frac{1}{4\sigma^2} - \frac{x^2}{8\sigma^4} \right) |\psi(x)|^2 dx}{\int |\psi(x)|^2 dx}$$

Metropolis Sampling

$$\langle f \rangle = \frac{\int p(x) f(x)\, dx}{\int p(x)\, dx}$$

$f(x)$: physical quantity; $p(x) = |\psi(x)|^2$

Consider the hamiltonian : $\mathcal{H} = -\frac{1}{2}\nabla^2 + \frac{1}{2}x^2$

Consider a wavefunction (not necessarily the ground state) : $\psi(x) = \exp(-x^2/4\sigma^2)$

Useful exercise, since in this case <E_{pot}> and <E_{kin}> can be calculated also analytically:

$$\langle E_{pot} \rangle = \frac{\langle \psi | \frac{1}{2}x^2 | \psi \rangle}{\langle \psi | \psi \rangle} = \frac{\int \frac{1}{2}x^2 |\psi(x)|^2 dx}{\int |\psi(x)|^2 dx} = \frac{1}{2}\sigma^2$$

$$\langle E_{kin} \rangle = \frac{\langle \psi | -\frac{1}{2}\nabla^2 | \psi \rangle}{\langle \psi | \psi \rangle} = \frac{\int \left(\frac{1}{4\sigma^2} - \frac{x^2}{8\sigma^4} \right) |\psi(x)|^2 dx}{\int |\psi(x)|^2 dx} = \frac{1}{8\sigma^2}$$

```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! metropolis_sampling.f90
!
! METROPOLIS sampling of physical observables for the hamiltonian:
! h= -1/2\nabla^2 + x^2/2 on psi^2(x), with psi(x)=exp(-x^2/(4\sigma^2))
... start from a given x=x0...
 do i=1,n
```

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

    endif
 enddo
```

```
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! metropolis_sampling.f90
!
! METROPOLIS sampling of physical observables for the hamiltonian:
! h= -1/2\nabla^2 + x^2/2 on psi^2(x), with psi(x)=exp(-x^2/(4\sigma^2))
... start from a given x=x0...
 do i=1,n
     ekin = ekin - 0.5_dp * ((x/(2*sigma**2))**2 - 1/(2*sigma**2))
     epot = epot + 0.5_dp * x**2
     etot = ekin + epot
     x1 = x1 + x
     x2 = x2 + x**2
     x3 = x3 + x**3
     x4 = x4 + x**4
     !ccccccccccccccccccccccccccc
     expx = - x**2 /(2*sigma**2)    !
     call random_number(rnd)        !
     xp = x + delta * (rnd-0.5_dp)  !
     expxp = - xp**2 /(2*sigma**2)  !   metropolis
     p = exp (expxp-expx)           !   algorithm
     call random_number(rnd)        !
     if (p > rnd) then              !
        x = xp                      !
     !ccccccccccccccccccccccccccc
        acc=acc+1.0_dp
     endif
 enddo
```

data accumulation on all points!

Correlations

Consider a random sequence of configurations.

How many Monte Carlo steps are required between two configurations to be considered uncorrelated?
=> study the autocorrelation function:

$$C(j) = \frac{\langle x_{i+j} x_i \rangle - \langle x_i \rangle^2}{\langle x_i^2 \rangle - \langle x_i \rangle^2}$$

where:

$\langle \ldots \rangle$: average over the random sequence (index i)

$C(j = 0) = 1$

$C(j \neq 0) = 0$ expected for totally uncorrelated points, since in that case $\langle x_i x_j \rangle = \langle x_i \rangle \langle x_j \rangle = \langle x_i \rangle^2$

Correlations

Consider a random sequence of configurations.

How many Monte Carlo steps are required between two configurations to be considered uncorrelated?
=> study the autocorrelation function:

$$C(j) = \frac{\langle x_{i+j} x_i \rangle - \langle x_i \rangle^2}{\langle x_i^2 \rangle - \langle x_i \rangle^2}$$

It is not always the case, but at least for *ergodic* simulations we should expect that the autocorrelation function approaches 0 as j $\rightarrow \infty$.

$C(j \neq 0) = 0$ expected for totally uncorrelated points, since in that case $\langle x_i x_j \rangle = \langle x_i \rangle \langle x_j \rangle = \langle x_i \rangle^2$

Origin of correlations

Metropolis algorithm:
necessarily the points of the walker are correlated each other over a short "time" scale (measured in terms of Monte Carlo steps; at least 1 time step!)

Correlation exponentially decaying with a certain characteristic "time" τ

Only points separated by 2τ or 3τ can be considered statistically independent

Calculation of correlations

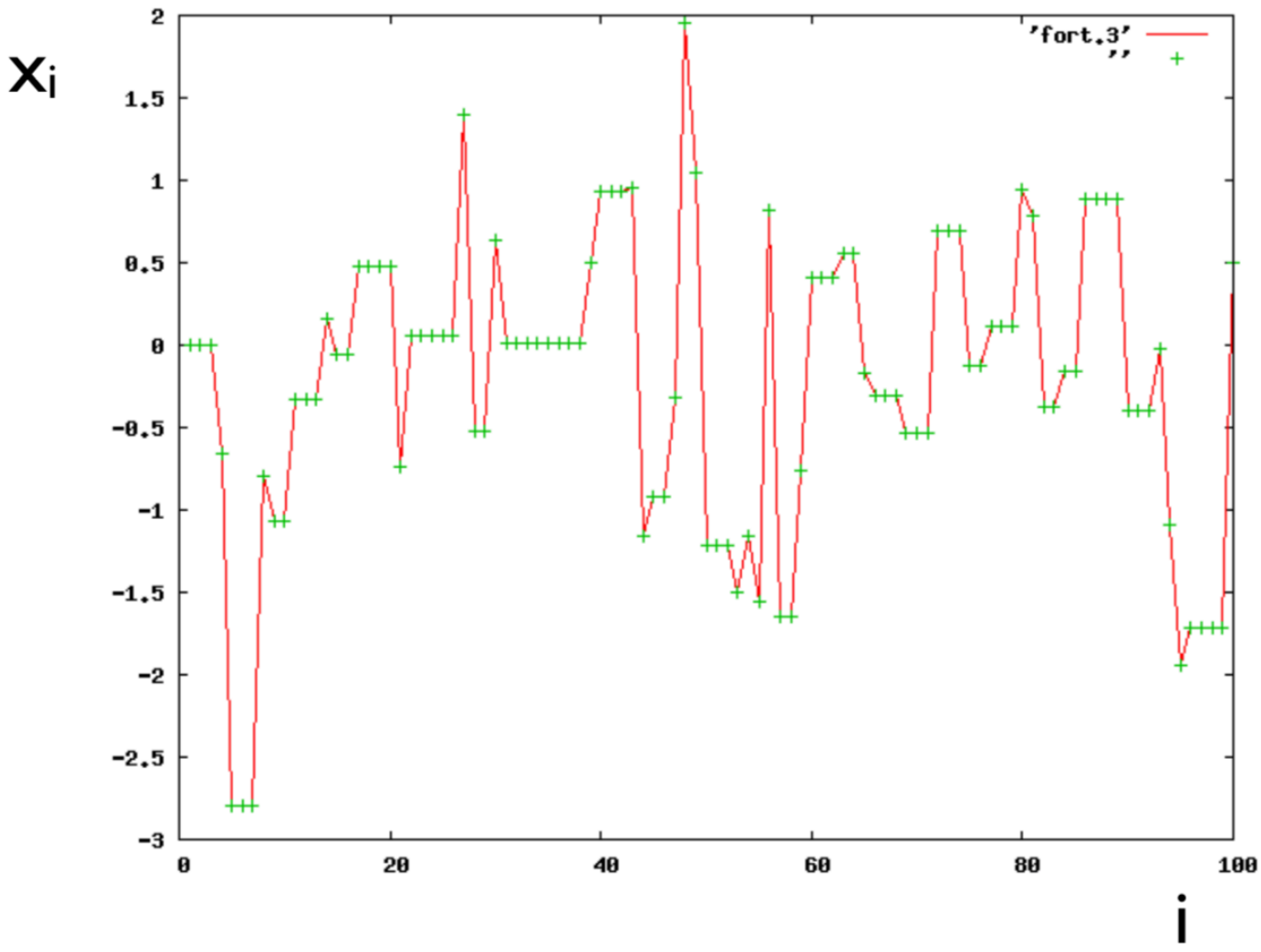
In Metropolis simulations, the autocorrelation time is often measured when the simulation is running:

- Create an array, say "corr" with j elements and initialize it to zero.
- Maintain a list of the j most recently computed values of the observable C. This can be an array of length j in which the value of C_n is stored at index "n mod j".
- At each step $n \geq j$ accumulate the values of $C_{n-j}C_{n-j+i}$ for $i = 0, 1, \ldots j-1$ in the array elements corr[i].
- At the end of the run, divide each corr[i] by $n - j$, subtract $\langle C \rangle^2$, and divide by corr[0] to normalize.

(do the code yourself!)

Calculate $C(j)$ for different j and check when it approaches 0

example of a sequence of points randomly (Gaussian) distributed generated according Metropolis



by its nature, Metropolis method introduces (at least short-range) correlations

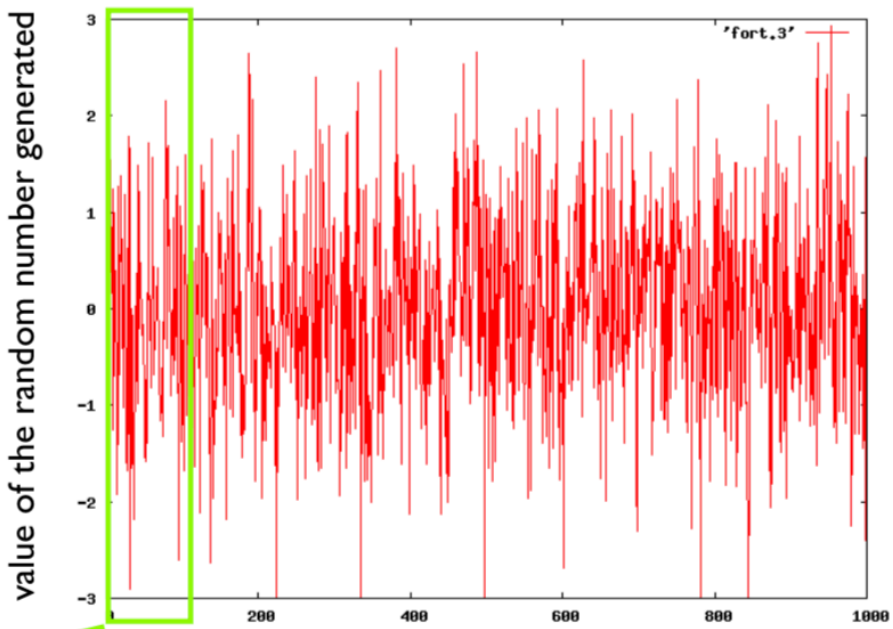
ex.VII.3 - Correlations

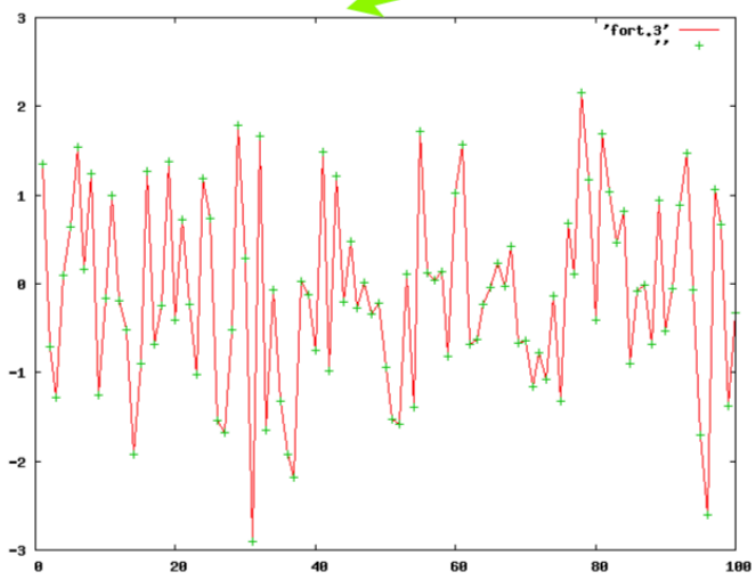
[a very simple calculation if the array points(n) is stored]

```
do j = 0, jmax

si=0._dp ; si2=0._dp ; sij=0._dp

  do i = 1,n-j
    si = si + points(i)
    si2 = si2 + points(i)**2
    sij = sij + points(i)*points(i+j)
  end do
    si = si/(n-j)
    si2 = si2/(n-j)
    sij = sij/(n-j)

write(2,*), j, (sij-si**2)/(si2-si**2)

end do
```

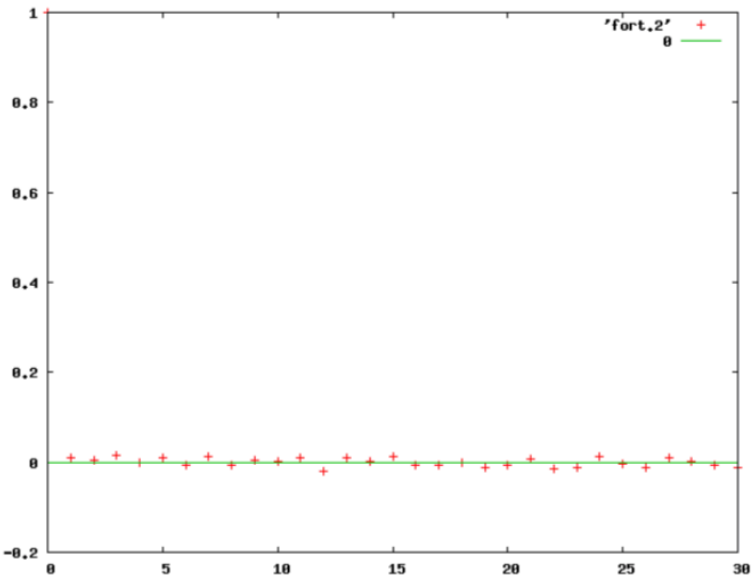
$$C(j) = \frac{\langle x_{i+j} x_i \rangle - \langle x_i \rangle^2}{\langle x_i^2 \rangle - \langle x_i \rangle^2}$$

Correlations - Box-Muller algorithm

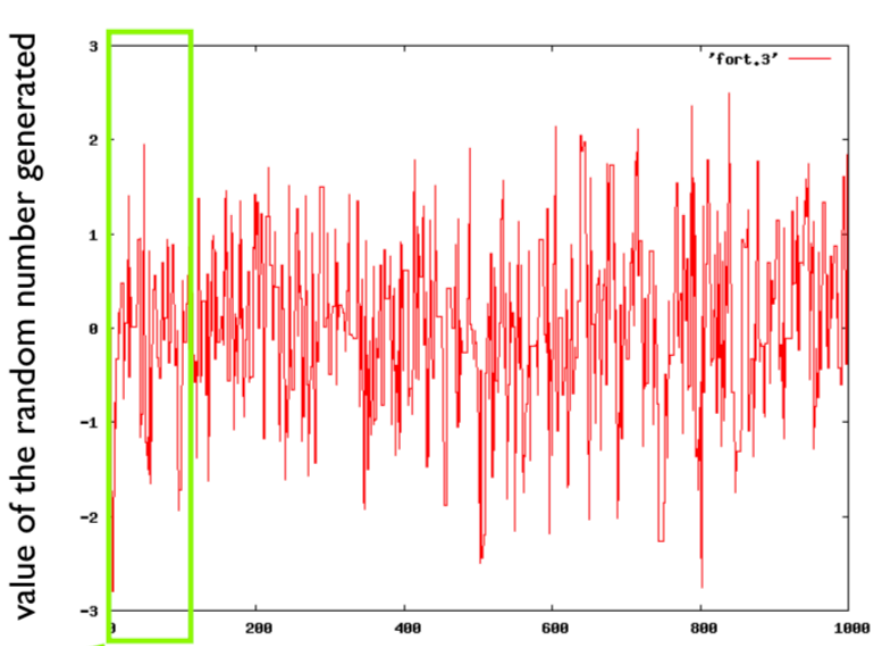


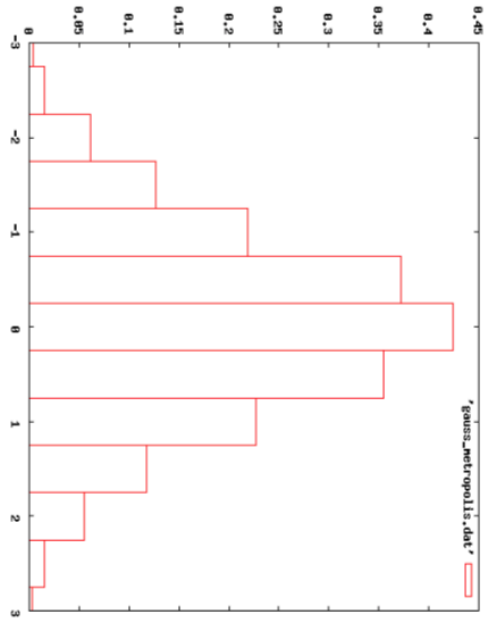


no correlations



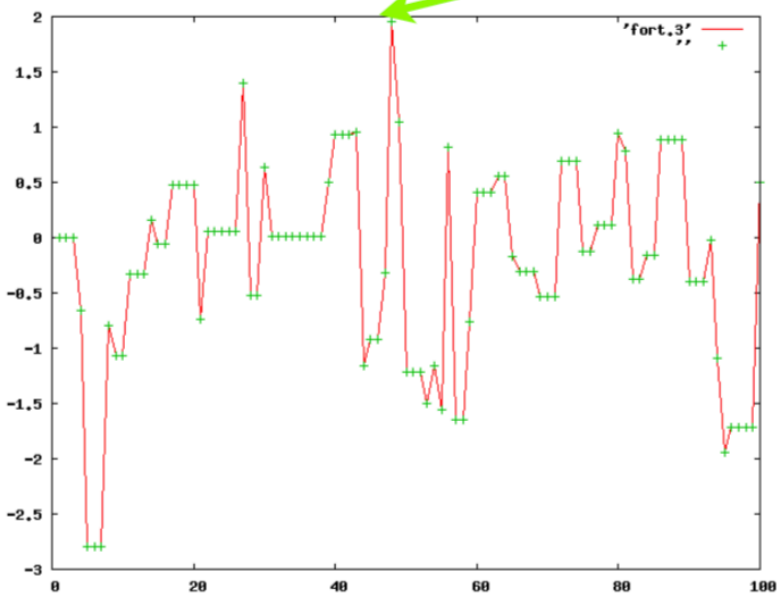
Correlations - Metropolis algorithm



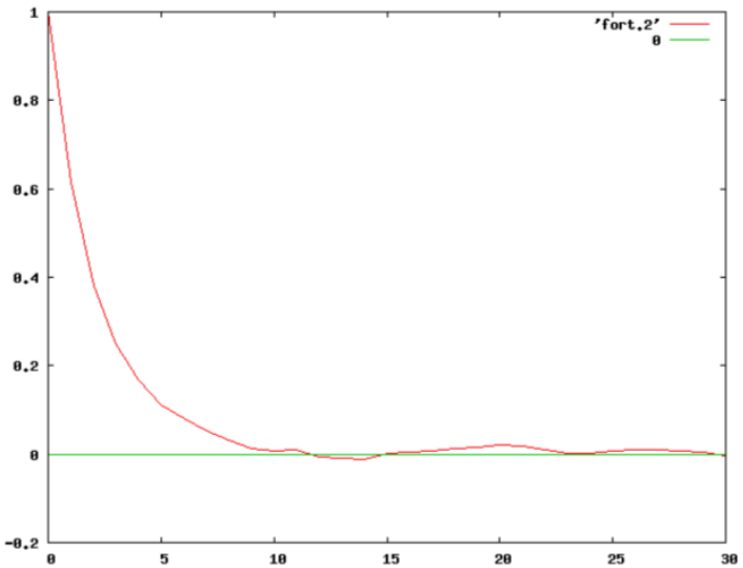


the first 1000 points

histogram over 10000 points



the short-term behavior of the Markov chain explains the correlations



Notes - I

* Correlation may cause fictitiously a variance of the averages much smaller than the actual error!

* The relationship

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

is based on the assumption of **uncorrelated** data (at least, for the block average, uncorrelated among different blocks)

Notes - II

Not good to use correlated points...

How to estimate correlations? How to estimate τ?
How to control the reliability of the statistical sampling?

Use **block averages** with different block size and compare the numerical error estimates $\sigma_s/\sqrt{s}$ and $\sigma_{s'}/\sqrt{s'}$.
If they coincides, the correlation time is smaller than the smallest block size used.

Suppose n=1000 data.
1) do blocking of s=20 sets with 50 points and calculate averages and errors
2) do blocking of s'=10 sets of 100 points and calculate averages and errors
50 is therefore the smallest block size used.
If $\sigma_{20}/\sqrt{20} \approx \sigma_{10}/\sqrt{10}$, this means that $\tau << 50$

Metropolis Sampling

Using a method to generate a distribution $p(x)$, we can efficiently sample integrals of the form

$$\langle f \rangle = \frac{\int p(x) f(x)\, dx}{\int p(x)\, dx}$$

Particularly useful integrals (or averages) are those related to **ensemble averages**

Metropolis algorithm

1) to generate random points with a given distribution

2) to calculate averages with importance sampling

3) in particular: in the canonical ensemble

Review of some concepts of statistical mechanics

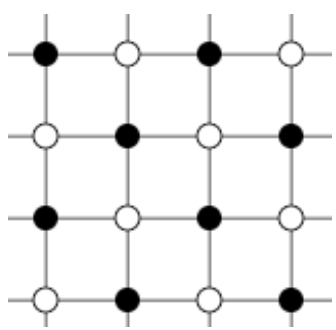
(microstate / macrostate / trajectory / statistical ensemble; statistical and temporal averages)

Microstates

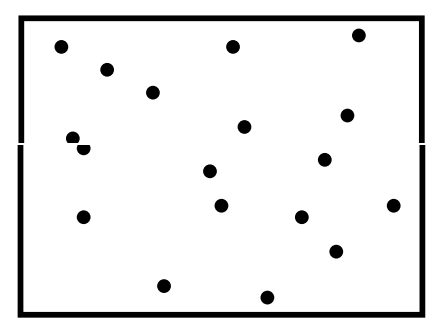
1) Examples of a **microstate:** characterized by:

- $|\Psi_\nu\rangle$ (ν index related to N particles) in the Hilbert space in Quantum Mechanics
- a point in the phase space (large number of variables) in Classical Mechanics

examples of microstates:



distribution of spins on a lattice
(open circles: spin up;
closed circles: spin down)



distribution of particles in a box
(list of positions)

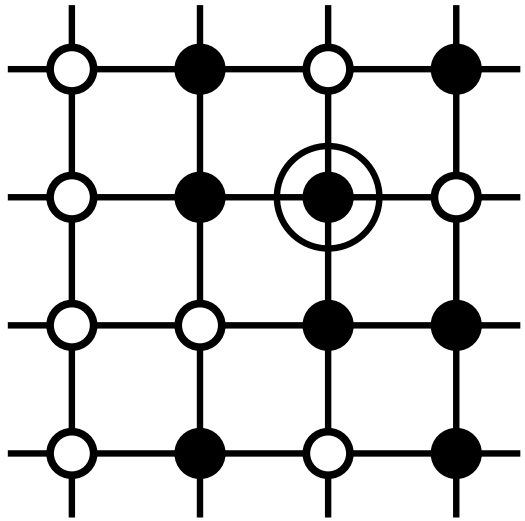
Macrostate and Trajectory

2) For a classical system, the temporal evolution (with possible changes of microstates) is a **trajectory** (a line) in the phase space:

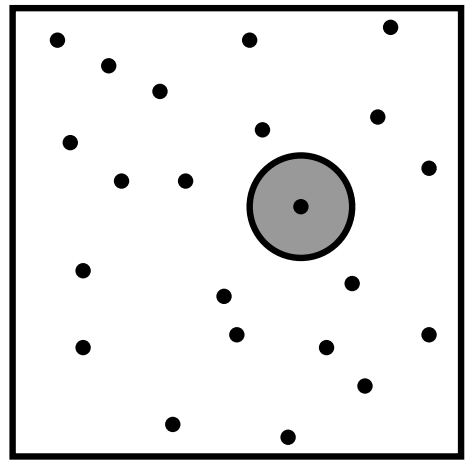
- along the trajectory, some parameters or variables such as N, or V, or T .... are fixed (constraints - **macrostate**)
- others do change

The trajectory is on a certain **surface** in the phase space (typically still high-dimensional), determined by the constraints

examples of changes of microstates:



flip of one spin in a lattice
(open circles: spin up;
closed circles: spin down)



moving one particle in a box

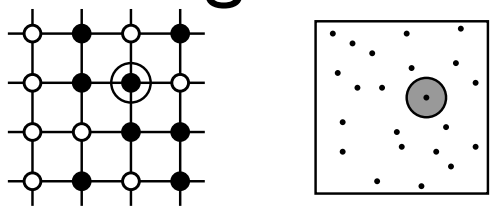
(the macrostate here is
characterized by the
temperature T and
by the total number of lattice sites)

(the macrostate here is
characterized by the
number of particles N)

(random choice of spin flip/particle move in case of Markov chain)

Stochastic processes and Markov chains

Stochastic process:
evolves through a series of well defined configurations (microstates in a given ensemble)
$C=\{C_1, C_2, \ldots C_N\}$ stochastically generated, i.e. C_{k+1} is obtained from the previous one, C_k, by making some random changes on the former.



Markov chain:
is defined by a matrix of transition probabilities T_{ij} from configuration i to j, and the probability $T_{k,k+1}$ for a transition from C_k to C_{k+1} does not depend on the previous history, i.e. no memory.

Statistical averages

3) which info from the **trajectory** in the phase space?

After a sufficiently long time, the system will assume all possible microstates compatible with the constraints i.e. with the macrostate : the ensemble of such microstates is a **statistical ensemble**

Suppose to make N independent measurements of an observable G:

$$G_{obs} = \frac{1}{N}\sum_{a=1}^{N} G_a$$ ← the measure is on a microstate

$$= \sum_{s} \underbrace{\left[\frac{1}{N} \cdot \left(\begin{array}{c}\text{\# of times in which the} \\ \text{microstate } s \text{ is observed}\end{array}\right)\right]}_{P_s} G_s$$

(M is the number of the possible different microstates s)

$$= \sum_{s=1}^{M} P_s G_s = < G >$$ **statistical average** or **ensemble average**

(P_s depends of the microstate s but also on the macrostate)

statistical average = temporal average

is a fundamental assumption of the statistical mechanics;
OK if:

- the system is **ergodic** (after a sufficient long time, the trajectory visits **all** the possible microstates)
- observation time is **long** $(T >> \tau_{relax\ or\ equil})$
- observations are **independent** $(d >> d_{correl})$

The canonical ensemble

(N,V,T) fixed. The probability that the system is in the microstate s with energy E_s is given by:

(here the energy identifies different microstates, it is not a characteristic of the macrostate)

$$P_s = \frac{1}{Z} e^{-\beta E_s}, \text{(canonical distribution)}$$

where $\beta = 1/kT$, and Z is a normalization constant. Because $\sum P_s = 1$,

$$Z = \sum_{s=1}^{M} e^{-E_s/kT}$$

partition function

(M: all accessible microstates of the system) characterized by different E_s

Ensemble averages (e.g. for the energy):

$$\langle E \rangle = \sum_{s=1}^{M} E_s P_s = \frac{1}{Z} \sum_{s=1}^{M} E_s e^{-\beta E_s}$$

Averages in the canonical ensemble

We can generate only a finite number m of the total number M of accessible microstates; we hope that:

$$\langle A \rangle \approx A_m = \frac{\sum_{s=1}^{m} A_s \, e^{-\beta E_s}}{\sum_{s=1}^{m} e^{-\beta E_s}} \qquad (*)$$

(Note: m, not M !)

<u>A crude MonteCarlo procedure</u>:

generate a microstate s at random, calculate E_s, A_s, and $e^{-\beta E_s}$, and evaluate (*)

Poorly efficient! P_s for the generated microstates could be rather small. An <u>importance sampling</u> method is better!

Importance sampling in the canonical ensemble

$$< A > \approx A_m = \frac{\sum_{s=1}^{m} A_s e^{-\beta E_s}}{\sum_{s=1}^{m} e^{-\beta E_s}} = \sum_{s=1}^{m} A_s \pi_s \quad \text{with} \quad \pi_s = \frac{e^{-\beta E_s}}{\sum_{s=1}^{m} e^{-\beta E_s}}$$

(*) NO importance sampling:
generate m configurations
with uniform random distribution,
then make a weighted average of A)

(Note : $\pi_s \neq P_s$!)

If we generate microstates according to π_s, the calculation of A_m reduces to:

$$A_m = \frac{1}{m} \sum_{s=1}^{m} A_s$$

with importance sampling

(generate m configurations
with random distribution π_s,
then make a simple average of A)

(much more efficient than (*))

Importance sampling in the canonical ensemble

Therefore, summarising: we aim at calculating

$$< A > \approx \frac{1}{m} \sum_{s=1}^{m} A_s \text{ with microstates } s \text{ generated according to } \pi_s$$

The transition matrix that generates microstates s according to π_s is :

$$T_{old,new} = min\left[1, \frac{\pi_{new}}{\pi_{old}}\right] = min\left[1, \frac{p_{new}}{p_{old}}\right] = min\left[1, \frac{e^{-\beta E_{new}}}{e^{-\beta E_{old}}}\right]$$

hence, using Metropolis, the procedure is as following:

Metropolis algorithm in the canonical ensemble

1. Establish an initial microstate.
2. Make a random trial change in the microstate. For example, choose a spin at random and flip it. Or choose a particle at random and displace it a random distance.
3. Compute $\Delta E \equiv E_{\text{trial}} - E_{\text{old}}$, the change in the energy of the system due to the trial change.
4. If ΔE is less than or equal to zero, accept the new microstate and go to step 8.
5. If ΔE is positive, compute the quantity $w = e^{-\beta \Delta E}$.
6. Generate a random number r in the unit interval.
7. If $r \leq w$, accept the new microstate; otherwise retain the previous microstate.
8. Determine the value of the desired physical quantities.
9. Repeat steps (2) through (8) to obtain a sufficient number of microstates.
10. Periodically compute averages over microstates.

Metropolis algorithm in the canonical ensemble

Steps 2 to 7 give really the desired distribution using:

$$T(i \to j) = \min(1, e^{-\beta \Delta E}) \quad \text{(Metropolis algorithm)},$$

where $\Delta E = E_j - E_i$.

A few remarks:

1) ERGODICITY implicitly assumed!

2) TEMPERATURE:

If $E_B > E_A$, accept the new (higher energy) configuration with probability $p = e^{-(E_B - E_A)/k_B T}$ This means that when the temperature is high, we don't mind taking steps in the "wrong" direction, but as the temperature is lowered, we are forced to settle into the lowest configuration we can find in our neighborhood.

Metropolis algorithm in the canonical ensemble: other remarks

1) Because it is necessary to evaluate only the ratio $P_j/P_i = e^{-\beta\Delta E}$, it is not necessary to normalize the probability.

$(P_j/P_i = \pi_j/\pi_i)$

2) Other choices of π_s are possible. Instead of writing:

$$< A > \approx A_m = \frac{\sum_{s=1}^{m} A_s e^{-\beta E_s}}{\sum_{s=1}^{m} e^{-\beta E_s}} = \sum_{s=1}^{m} A_s \pi_s = \frac{1}{m} \sum_{s=1}^{m} A_s$$

(no importance sampling) (with importance sampling)

rewrite:

$$A_m = \frac{\sum_{s=1}^{m} (A_s/\pi_s)\, e^{-\beta E_s} \pi_s}{\sum_{s=1}^{m} (1/\pi_s)\, e^{-\beta E_s} \pi_s} .(\text{no importance sampling})$$

If we generate microstates with probability π_s, eq. becomes:

$$A_m = \frac{\sum_{s=1}^{m} (A_s/\pi_s)\, e^{-\beta E_s}}{\sum_{s=1}^{m} (1/\pi_s)\, e^{-\beta E_s}} .(\text{importance sampling})$$

3) If $T(i \to j)e^{-\beta E_i} = T(j \to i)e^{-\beta E_j}$ (detailed balance),
Metropolis algorithm generates states with Boltzmann distribution

(we will prove it empirically : see exercise 4)

Some programs:

on
$/home/peressi/comp-phys/VII-metropolis/
[do: $cp /home/peressi/.../VII-metropolis/* .]
or in moodle2.units.it

gauss_metropolis.f90

metropolis_sampling.f90
direct_sampling.f90
boltzmann_metropolis.f90

Boltzmann distribution in the canonical ensemble

The Metropolis algorithm really produces microstates with the Boltzmann distribution:
application to ideal classical 1D gas (exercise n. 4)

1 free particle: Energy: $E = \frac{1}{2}mv^2$
in this case, velocity or energy labels a microstate

(the energy with a factor of 2 , due to +/- sign of v);

we generate different microstates by random variations of the velocity and we accept/reject with Metropolis

Important quantities are the probabilities:
P(v)dv that the system has a velocity between v and v+dv
or P(E)dE that the system has an energy between E and E+dE

ideal classical 1D gas

A particle moving randomly has in each direction a distribution of the component of the velocity:

$$f(v_x) = \left(\frac{m}{2\pi k_B T}\right)^{1/2} e^{-mv_x^2/2k_B T} \tag{1}$$

$$\langle v_x^2 \rangle = \int_{-\infty}^{+\infty} v_x^2 f(v_x) dv_x = \frac{k_B T}{m} \tag{2}$$

In 1D:

$$f(v)2dv = P(E)dE$$

that gives:

$$P(E) = \frac{1}{(\pi k_B T)^{1/2}} \frac{1}{\sqrt{E}} e^{-E/k_B T}$$

In 3D:

$$P(E) = \frac{2}{\sqrt{\pi}} \frac{1}{(k_B T)^{3/2}} \sqrt{E} \exp\left(-\frac{E}{k_B T}\right) \qquad \text{(3D)}$$

Boltzmann distribution in the canonical ensemble

```
 # T        :  1.00000
  # <E0>     :  .000000
  # <v0>     :  .000000
  # dvmax    :  2.00000
  # deltaE   :  5.000000E-02
  # nbin     :  79

==> boltzmann.1K <==
  # nMCsteps:  1000
  # <E>      :  .501263
  # <v>      :  7.456664E-02
  # accept.  :  .692000
  # sigma    :  .713780

==> boltzmann.10K <==
  # nMCsteps:  10000
  # <E>      :  .507580
  # <v>      :  3.366172E-02
  # accept.  :  .707700
  # sigma    :  .726145

==> boltzmann.1M <==
  # nMCsteps:  1000000
  # <E>      :  .500138
  # <v>      :  1.833840E-04
  # accept.  :  .693837
  # sigma    :  .707472
```

$$T = 1 \rightarrow \langle E \rangle (expected) = 0.5 \qquad (m = 1)$$

$$\sigma/\sqrt{n} = 0.022 \qquad (\sigma \text{ is the variance of the energy})$$

$$\sigma/\sqrt{n} = 0.007$$

$$\sigma/\sqrt{n} = 0.0007$$

```
NOTE:
- Accuracy of ~ 1% on <E> and 10% on <v> :  NMCS=1000 is enough
- NOT ENOUGH to well reproduce the BOLTZMANN DISTRIBUTION! (1M needed!)
- ACCEPTANCE RATIO: constant, depends only on dvmax
- SIGMA also
```

Boltzmann distribution in the canonical ensemble

many particles: Energy: $E = \sum_{i=1}^{N} \frac{1}{2} m_i v_i^2$

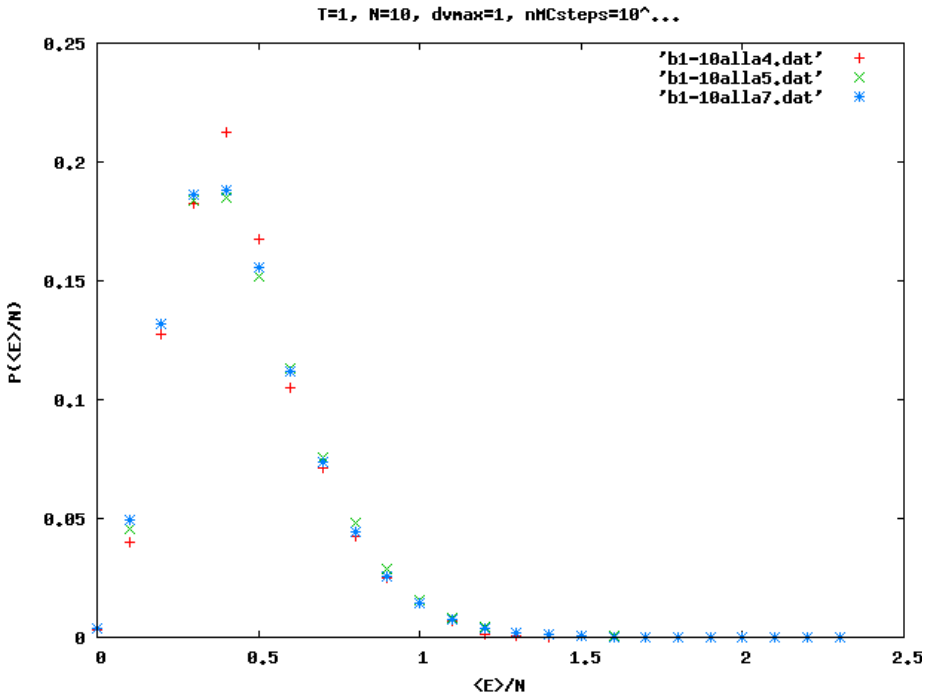
in this case, the energy is NOT a label of a microstate (there are several microstates with the same total energy)

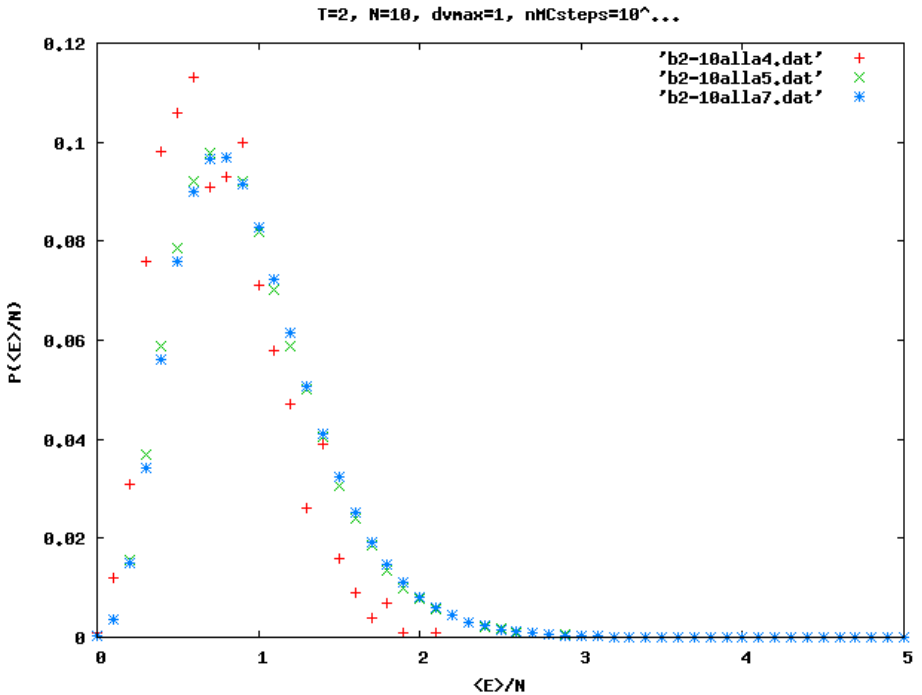
Note: the energy histogram is NOT the distribution of microstates!

$$P(E) = \sum_{\substack{states\ s \\ with\ E_s = E}} P_s \quad \text{with} \quad P_s = \frac{1}{Z} e^{-\beta E_s}$$

$$P(E) \propto= e^{\frac{-(E-\langle E \rangle)^2}{2\sigma^2}} \quad \text{with } \langle E \rangle \text{ average over all the microstates}$$

What is P(E)? (exercise n. 4)


T=1, N=10, dvmax=1, nMCsteps=10^...
'b1-10alla4.dat'
'b1-10alla5.dat'
'b1-10alla7.dat'
P(<E>/N)
<E>/N
0.25
0.2
0.15
0.1
0.05
0
0
0.5
1
1.5
2
2.5


T=2, N=10, dvmax=1, nMCsteps=10^...
'b2-10alla4.dat'
'b2-10alla5.dat'
'b2-10alla7.dat'
P(<E>/N)
<E>/N
0.12
0.1
0.08
0.06
0.04
0.02
0
0
1
2
3
4
5