



993SM - Laboratory of Computational Physics lecture 10 - part 1 May 13, 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

Lattice gas models

idea: recover the treatment of RWs
but the walkers now move together and interact

other topics:

- Macroscopic systems towards equilibrium
- Stochastic fluctuations
- Simulated annealing

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

Random Walks

Dependence of $\langle R^2(t) \rangle$ on t :

- **normal behavior:** $\langle R^2(t) \rangle \sim t$
for the brownian motion
- **superdiffusive behavior:** $\langle R^2(t) \rangle \sim t^{2\nu}$ with $\nu > 1/2$
in models where self-intersections are unfavored
- **subdiffusive behavior** $\langle R^2(t) \rangle \sim t^{2\nu}$ with $\nu < 1/2$
in models where self-intersections are favored

t (time) $\leftrightarrow N$ (number of steps); $t = N \Delta t$

$\langle \rangle$ = avg. over walkers

RW and diffusion

- consider the normal behaviour: $\langle R^2(t) \rangle \sim t$

The **quantity**:

$$D(t) = \frac{1}{2dt} \langle \Delta R(t)^2 \rangle$$

(where d is the dimensionality of the system)

should go asymptotically to a constant value for large t ,

the **autodiffusion coefficient**:

$$D = \lim_{t \rightarrow \infty} D(t)$$

For $d=1$: $\langle \Delta R_N^2 \rangle = N\ell^2$

$$D = \lim_{t \rightarrow \infty} \frac{1}{2dt} \langle \Delta R^2(t) \rangle = \frac{N\ell^2}{2t} = \frac{\ell^2}{2\Delta t}$$

$\langle \rangle$ = avg. over walkers

RW and diffusion in 1D

The probability that a RW of N steps (N large) ends at position x is given by:

$$P_N(x) = \sqrt{\frac{2}{\pi N}} \exp\left(-\frac{x^2}{2N}\right)$$

Considering that $t = N\Delta t$, defining $D = \frac{\ell^2}{2\Delta t}$, and measuring x in units of ℓ , we get:

$$P(x, t) = \sqrt{\frac{1}{\pi Dt}} \exp\left(-\frac{x^2}{4Dt}\right)$$

which is the fundamental solution of the diffusion equation, apart from a factor of 2 in the normalization due to the spatial discretization. The continuum solution is:

$$P(x, t) = \sqrt{\frac{1}{4\pi Dt}} \exp\left(-\frac{x^2}{4Dt}\right)$$

i.e., a Gaussian distribution with $\sigma^2 = 2Dt$ which describes a pulse gradually decreasing in height and broadening in width in such a manner that its area is conserved.

RW and diffusion on a 3D lattice

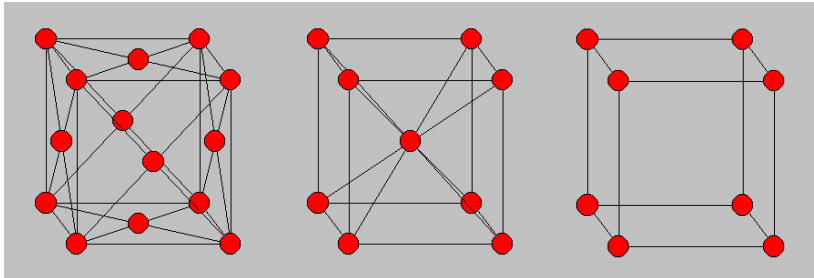
Discretized model:
if 1 step = move by 1 bond length,
we expect:

$$RMS = \sqrt{\langle R_N^2 \rangle} = \sqrt{N} d_{NN}$$

$$\text{fcc} : d_{NN} = a_0 \frac{\sqrt{2}}{2} = 0.71a_0$$

$$\text{bcc} : d_{NN} = a_0 \frac{\sqrt{3}}{2} = 0.87a_0$$

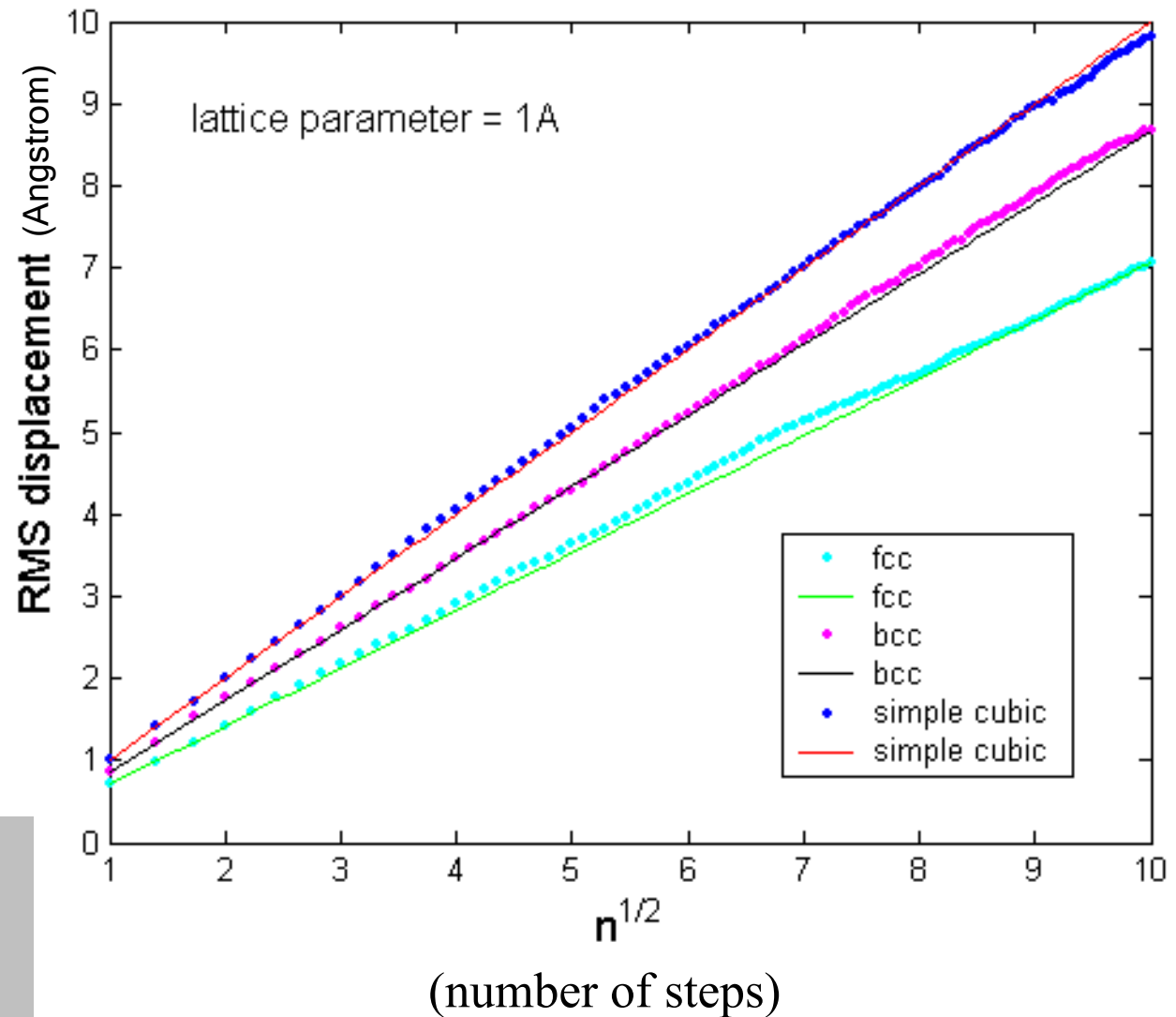
$$\text{sc} : d_{NN} = a_0$$



fcc

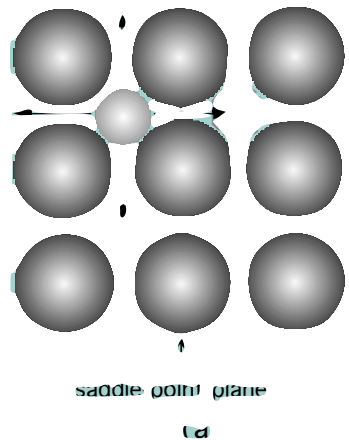
bcc

sc

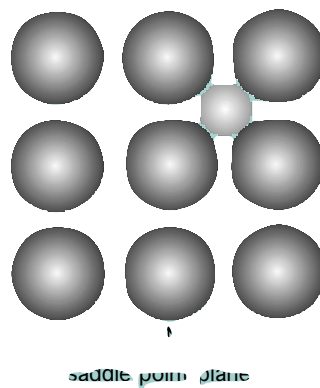
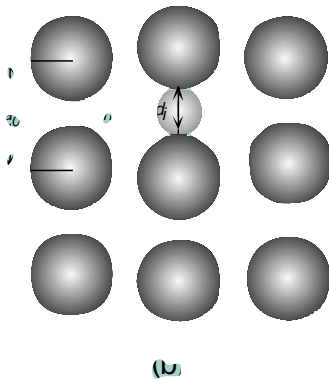


(=> D depends on the structure of the lattice)

Example of diffusion in solids

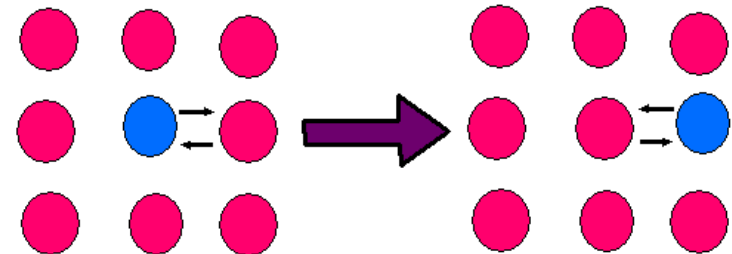


INTERSTITIAL IMPURITIES

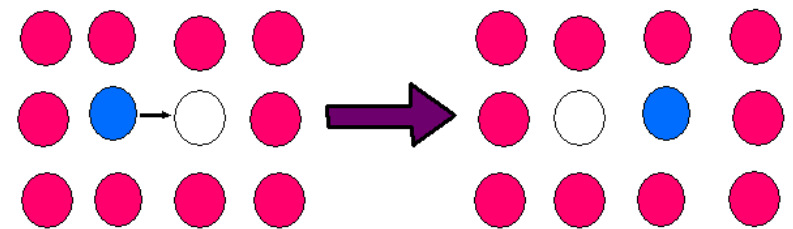


SUBSTITUTIONAL IMPURITIES

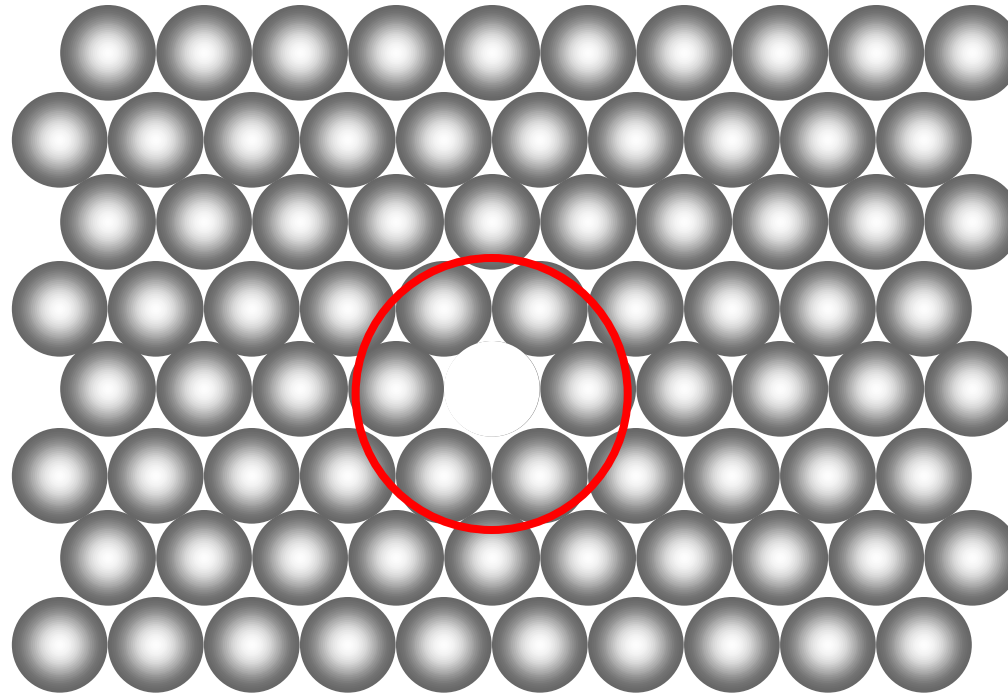
Direct exchange:



Vacancy assisted diffusion:



Vacancies diffusion in solids



vacancies themselves can diffuse!

... but typically:
more than one single interstitial,
more than one single impurity,
or more than one single vacancy...

**A SIMPLE RW MODEL
IS NOT ENOUGH!**

Lattice Gas model

interaction !

Consider a finite lattice with some density ρ of N_p particles. The particles can move on the lattice by jumps to the nearest sites, but two particles can not occupy the same site. This is a simple example of a **restricted** random walk (see above). The physical interpretation is e.g. vacancies moving in a lattice.

To simulate this kind of system, we need a bit more of an advanced approach than before. First of all, we need to simulate the motion of all the particles at the same time, not taking the average over many independent single-particle motions as was done before.

2D Lattice Gas model

- 1° Choose number of particles N_p , number of steps N_{steps} , side length L . Set Δt and lattice size a . (our old ℓ)
- 2° Set all positions in the $L \times L$ grid to be empty
- 3 a° Generate N_p particle coordinates randomly on the grid, checking that no two particles end up on the same points.
- 3 b° Mark the points with the particles in the $L \times L$ grid as filled.
- 4° Loop over MC steps of time Δt
 - 5° Loop from 1 to N_p
 - 6° Pick one particle i at random
 - 7° Find which positions it can jump to. If none, return to step 6° (*)
 - 8° Let the particle jump to one of the allowed directions j by a displacement $x_i = x_i + \delta x_j, y_i = y_i + \delta y_j$, enforce periodic boundaries on x and y
 - 9° Set $dx_i = dx_i + \delta x, dy_i = dy_i + \delta y$ (where periodic boundaries do not play a role!)
 - 10° End loop from 1 to N_p
 - 11° Update time $t = t + \Delta t$
- 12° End loop over MC steps
- 13° Output $\langle \Delta R^2 \rangle = \langle dx_i^2 + dy_i^2 \rangle$ and calculate diffusion coefficient $D(t) = \frac{1}{2dt} \langle \Delta R(t)^2 \rangle$

average over the particles

Lattice Gas model

(*) Different dynamics can be implemented, for instance:

- find which nearest neighbor sites are free and jump in one of them randomly chosen (if any) (this is actually mentioned in the previous slide and implemented in the code we are going to discuss) OR
- choose randomly one nearest neighbor site and jump only if it is free

Different dynamics => different behavior with concentration

Lattice Gas model

The crucial difference here to the previous random walk algorithms is that the outer loop goes over MC steps, the inner one over particles. When the walkers are independent of each other (“non-interacting”) we can deal with one walker at a time, saving memory since storage of all particles is not needed.

But here the walkers (the particles) are “interacting”

Programs:

on

`$/home/peressi/comp-phys/IX-latticegas-fractals/`

`[do: $cp /home/peressi/.../IX-latticegas-fractals/* .]`

or on moodle2

latticegas.f90

entropy.f90

box.f90

simulated_annealing.f90

Implementation of the model on 2D SQ lattice (latticegas.f90)

```
...
logical,allocatable::lattice(:,:) ! (occ./non occ.=.true./.false.)
integer,allocatable::x(:),y(:) ! instantaneous positions of Np labelled particles
double precision, allocatable :: dx(:),dy(:) ! displ. from the starting point
integer :: free(4),nfree ! occupation of nearest neighbors
integer :: dxtrial(4),dytrial(4) ! trial move (instantaneous displacements)
integer :: xnew(4),ynew(4) ! 4 new possible positions
.....
allocate(lattice(0:L-1,0:L-1))
allocate(x(Np),y(Np))
allocate(dx(Np),dy(Np))

...
lattice = .false. ! Mark all positions as empty

...
! Enumerate directions: 1=right; 2=left; 3=up; 4=down
dxtrial(1)=+1; dytrial(1)= 0;
dxtrial(2)=-1; dytrial(2)= 0;
dxtrial(3)= 0; dytrial(3)=+1;
dxtrial(4)= 0; dytrial(4)=-1;
```

```

! INIZIALIZE THE LATTICE : Generate Np particles on LxL lattice
do i=1,Np
  do ! Loop until empty position found, UNBOUNDED LOOP!
    call random_number(rnd)      !which has dimension(2)
    x(i)=int(rnd(1)*L)
    y(i)=int(rnd(2)*L)
    if (lattice(x(i),y(i))) then
      ! Position already filled, loop to find new trial
      cycle      !REMEMBER: JUMP AT THE END OF THIS LOOP (NOT EXIT)
    else
      lattice(x(i),y(i))=.true.
      ! Successful, place next particle
      exit
    endif
  enddo
  dx(i)=0.0d0; dy(i)=0.0d0;
enddo

```


! MONTE CARLO LOOP

```
do istep=0,Nsteps-1 ! Loop over MC steps
do istubstep=1,Np ! Move each particle once every MC step (on av.)
! Pick one particle at random
call random_number(rnd1)
i=int(rnd1*Np)+1 ! 1 =< i =< Np;
! Find possible directions (j=1,...,4) for moving, store them
in free() ... (NOTE: different possible recipes !!!)
! If no free positions, get a new particle ; otherwise choose
! one possible direction (j) and update (x,y) with (xnew,ynew):
.....
!Empty the old position and fill the new one:
lattice(x(i),y(i))=.false.
lattice(xnew(j),ynew(j))=.true.
enddo
t=t+deltat
enddo
```

Another fundamental part:
calculation of distance from initial pos. for each particle
(do not use PBC for that!),
accumulation of data...

```
! Get total displacement using dx,dy  
! dx,dy are individual displacements from the  
! starting point => these d*sum are summed  
! over time and particles
```

```
dxsum=0.0d0; dysum=0.0d0;
```

```
dxsqsum=0.0d0; dysqsum=0.0d0;
```

```
do i=1,Np
```

```
dxsum=dxsum+dx(i); dysum=dysum+dy(i);
```

```
dxsqsum=dxsqsum+dx(i)*dx(i);
```

```
dysqsum=dysqsum+dy(i)*dy(i);
```

```
enddo
```

```
print *, 'dxsum', dxsum, ' dysum', dysum
```

```
print *, 'dxsqsum', dxsqsum, ' dysqsum', dysqsum
```

Concentration dependent diffusion coefficient

And here is a series of results:

N_p	L	concentration N_p/L^2	D (cm ² /s)	
10	100	0.001	9.76	E-008
10	100	0.001	1.12	E-007
100	100	0.01	1.02	E-007
100	100	0.01	9.46	E-008
10000	1000	0.01	9.89	E-008
1000	100	0.1	9.11	E-008
1000	100	0.1	9.42	E-008
100000	1000	0.1	9.40	E-008
3000	100	0.3	8.28	E-008
3000	100	0.3	7.91	E-008
6000	100	0.6	5.89	E-008
6000	100	0.6	5.91	E-008
9000	100	0.9	1.77	E-008
9000	100	0.9	1.78	E-008
900000	1000	0.9	1.82	E-008
9900	100	0.99	1.83	E-009
9900	100	0.99	1.86	E-009

Here: 2d example

1 MC step = 1 ns

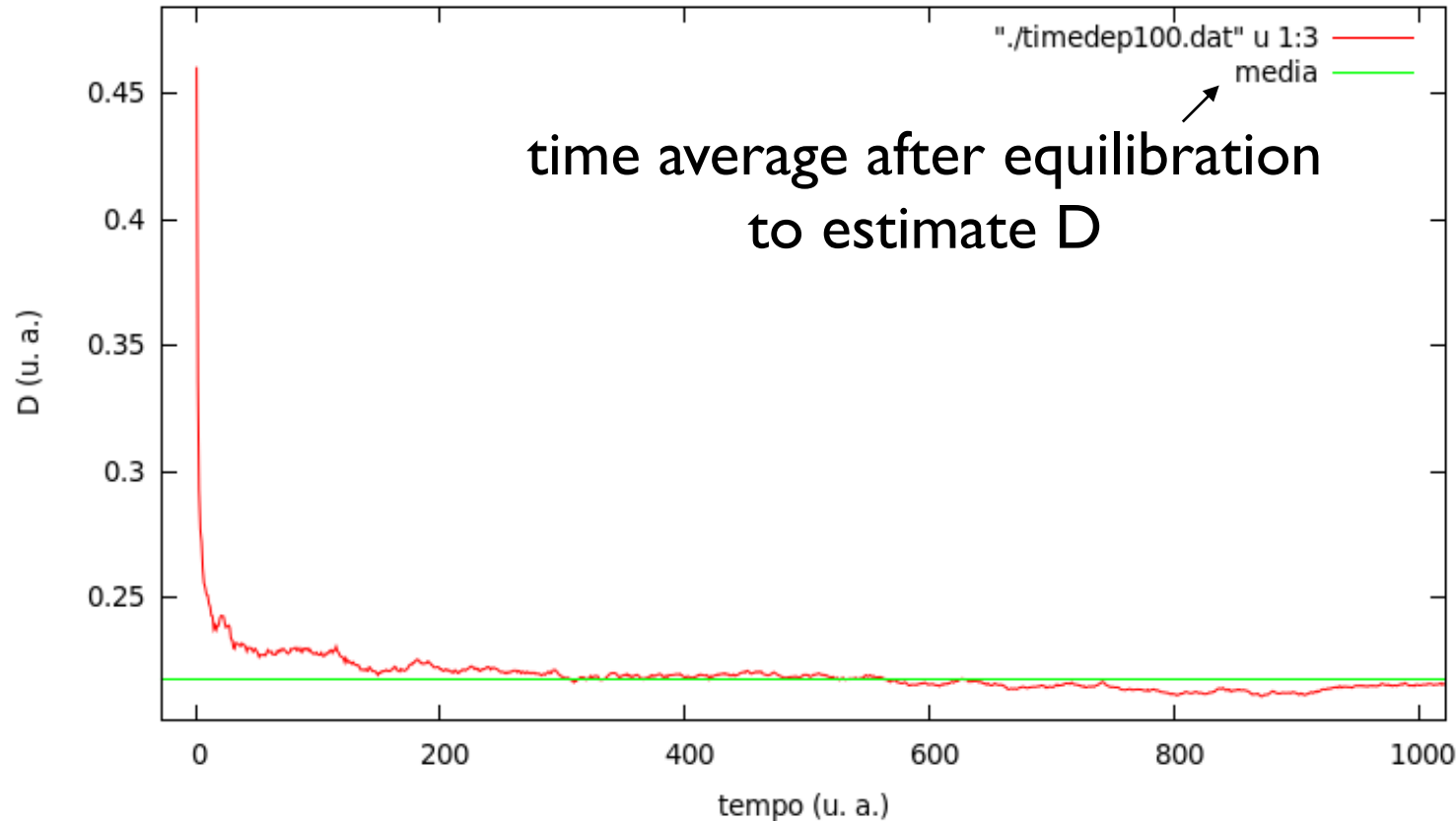
unit step length = 2 Å

What does this mean? At small concentrations, the system behaves essentially as an unconstrained random walk. For that one, we know that $\langle \Delta R^2 \rangle$ should be equal to $a^2 N$, where N is the number of steps, and a is the jump distance, and the result for the diffusion coefficient should be

$$D = \frac{\langle \Delta R^2 \rangle}{4t} = \frac{(2 \text{ \AA})^2 N}{4N \Delta t} = \frac{(2 \text{ \AA})^2}{4 \times 1 \text{ ns}} = 10^{-7} \frac{\text{cm}^2}{\text{s}}$$

Discussing Ex. I

(I.a) Study $D(t)$ for a fixed value of ρ , for instance 0.2. Although D is defined as the limit $t \rightarrow \infty$, it is instructive to follow $D(t)$ as a function of time: for this model, it fluctuates after a short equilibration time and no appreciable improvements in the statistics are achieved by increasing t .



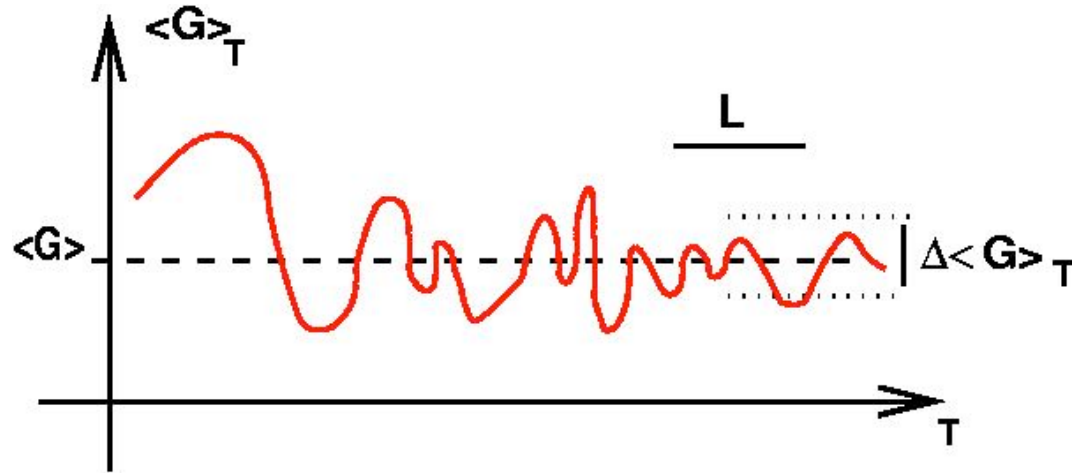
this is $D(t)$ (averaged over particles);
calculate it for $t \rightarrow \infty$

Averages

$$\langle G \rangle_T = \frac{1}{T} \sum_{t=1}^T G_t$$

Thermally equilibrated averages:

$$\langle G \rangle = \lim_{T \rightarrow \infty} \langle G \rangle_T$$



But in practice T is finite, and $\langle G \rangle_T$ oscillates (varying T):

divide T into intervals $A, B, C \dots$ of length L and sum **(block averages)**:

$$\langle G \rangle_T = \frac{L}{T} \sum_{I=A,B,C,\dots} \langle G \rangle^{(I)}$$

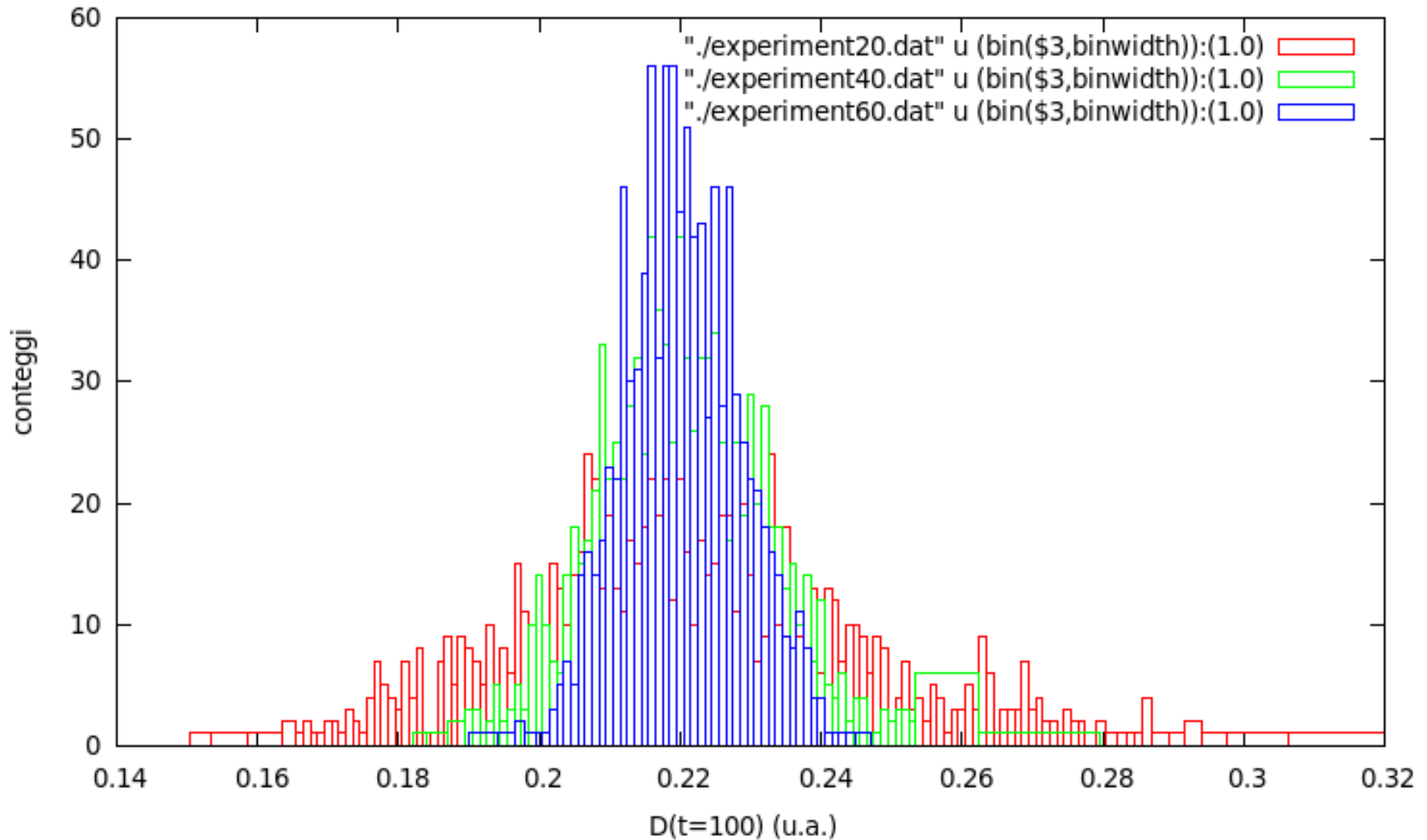
(σ_s/\sqrt{s})
 (σ_n/\sqrt{n})

$$\Delta \langle G \rangle_T = \left[\frac{L}{T} \sum_I \left(\langle (G^{(I)})^2 \rangle - (\langle G^{(I)} \rangle)^2 \right) \right]^{1/2} \xrightarrow{T \rightarrow \infty} \sim \frac{1}{\sqrt{T}}$$

Note: not always $\Delta \langle G \rangle_T$ is a good indicator of the actual error!
(remind "ergodicity")

A usually, we can estimate the statistical error associated to the estimate of D

(here: histogram done collecting data in the time evolution of $D(t)$)



size effect in the determination of D (concentration ρ fixed)!
(more later)

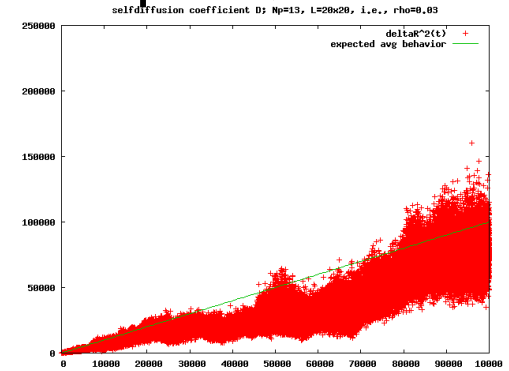
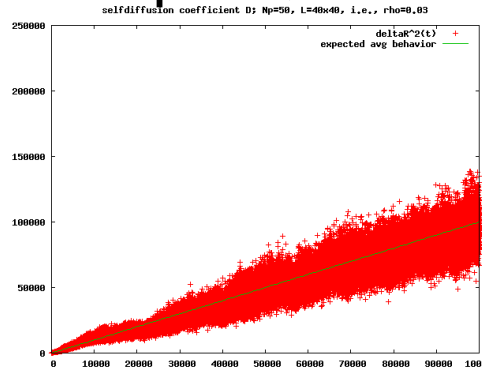
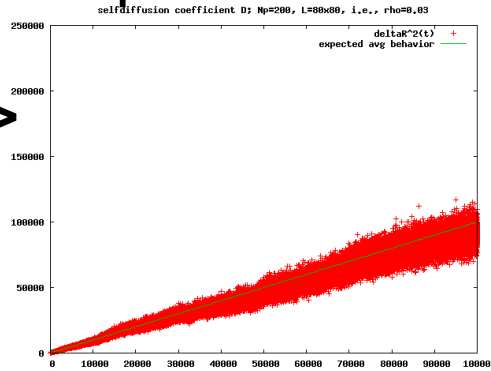
(I.I) ... Better statistics for D can be obtained by averaging D over as many particles as possible (i.e., for a given ρ)... Here $\rho=0.03$

$N_p=200, 80 \times 80$

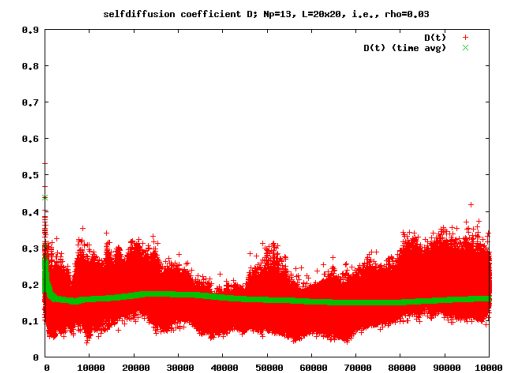
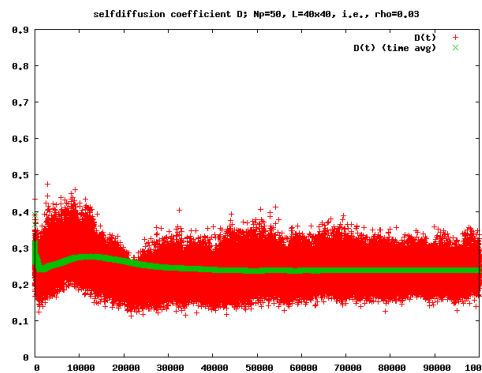
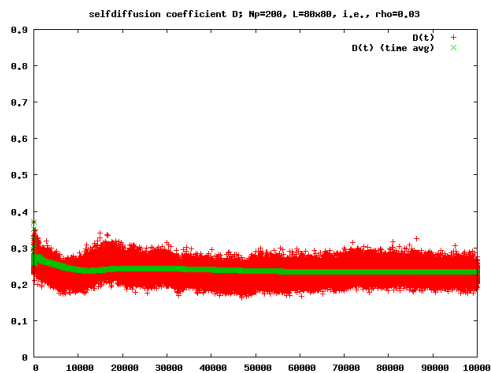
$N_p=50, 40 \times 40$

$N_p=13, 20 \times 20$

$\langle \Delta R^2(t) \rangle$
and
expected
behavior

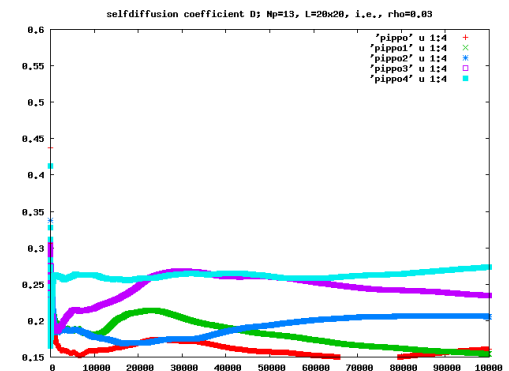
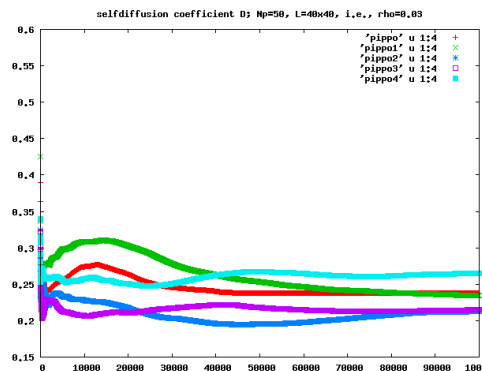
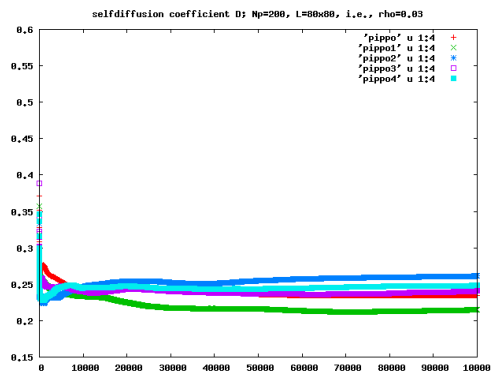


$D(t)$
and
 $\langle D(t) \rangle_t$



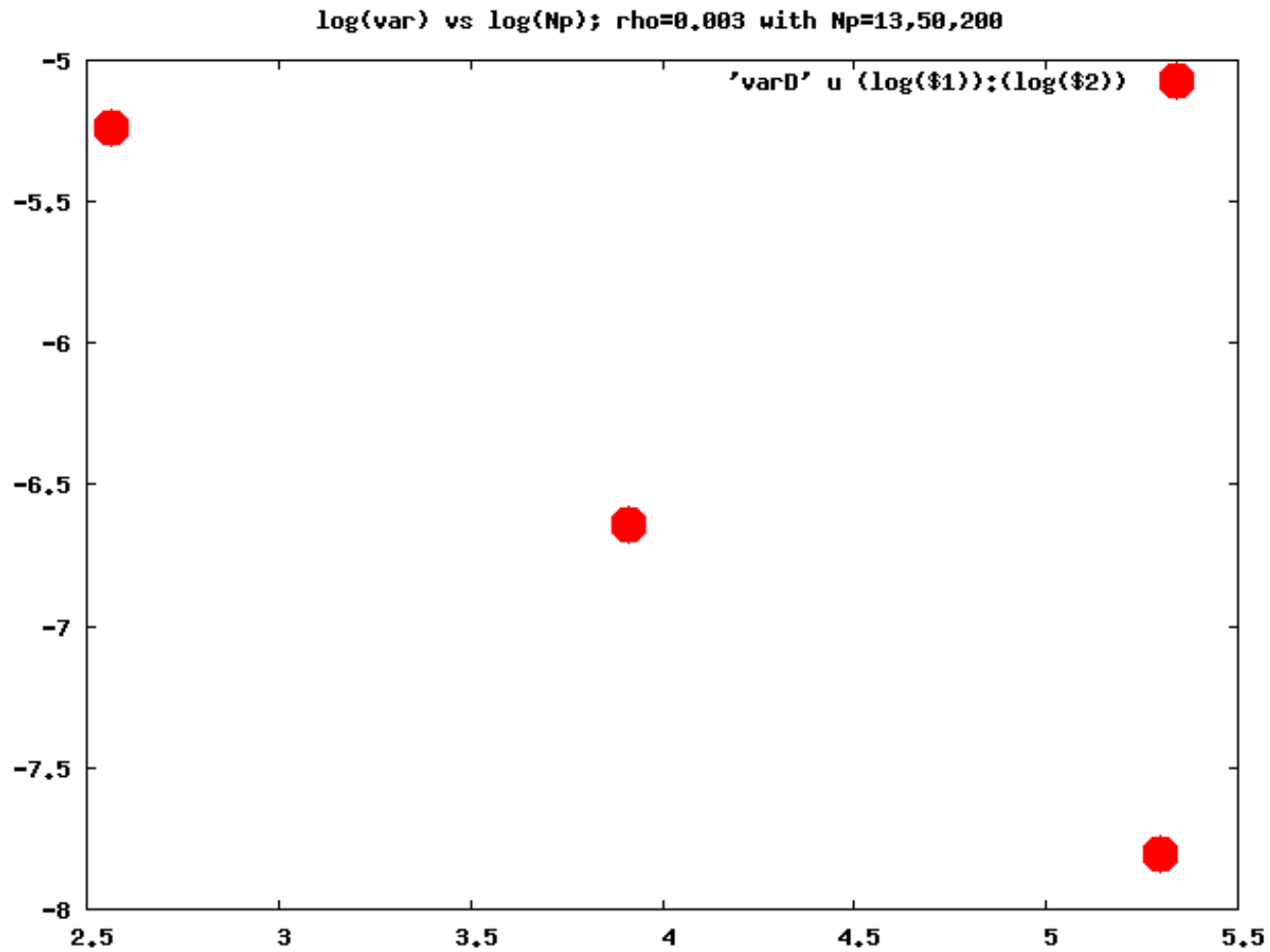
time averaged:
 $\langle D(t) \rangle = \int_0^t D(t') dt$

5 runs



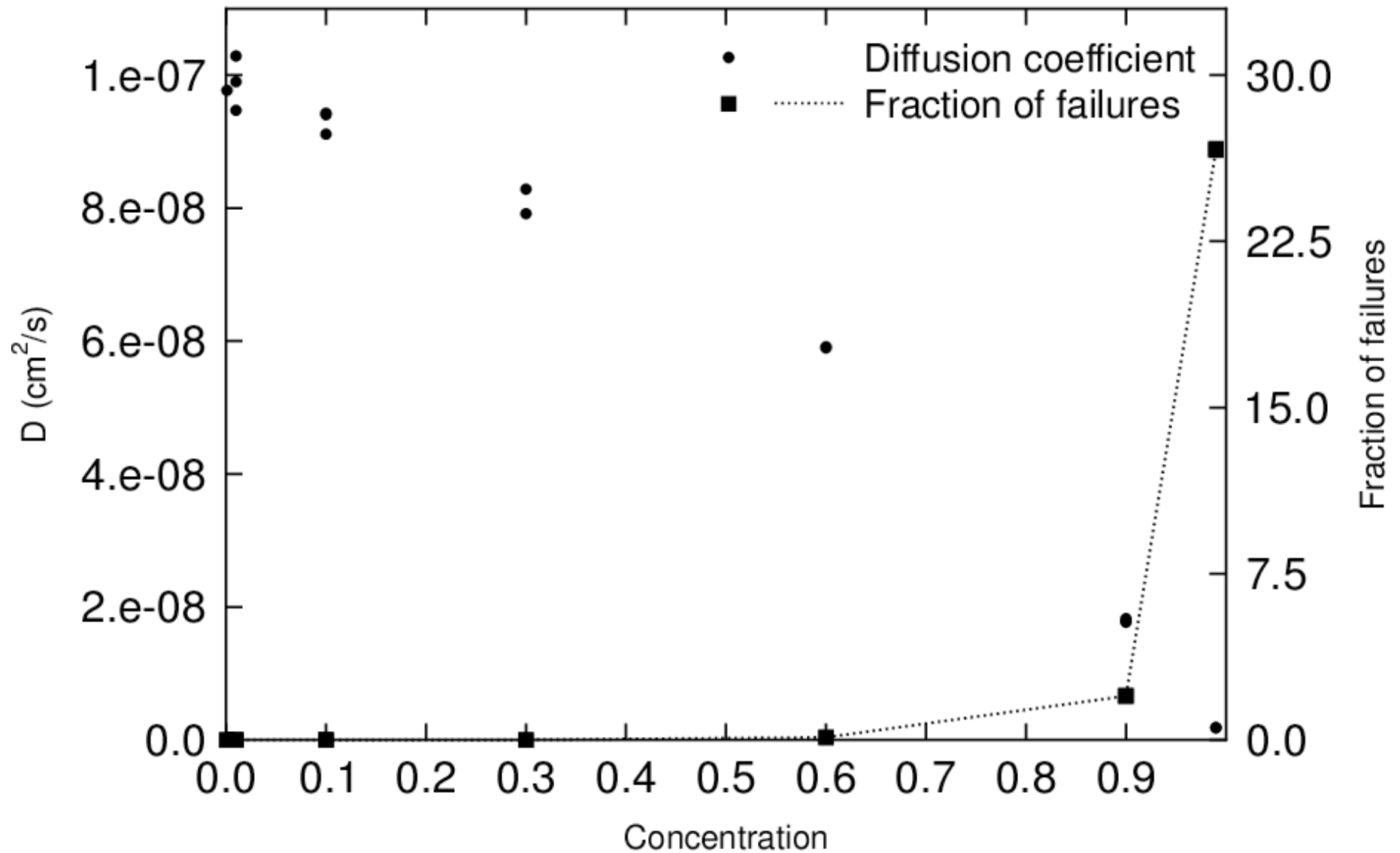
(we expect the limit of the simple 2D RW on a square lattice, with $D=0.25$)

Ex. 1 (...) Verify that deviations of $\langle D(t) \rangle_t$ from its mean value are proportional to the inverse square root of the total number of particles.



σ^2_D proportional to $1/N_p$

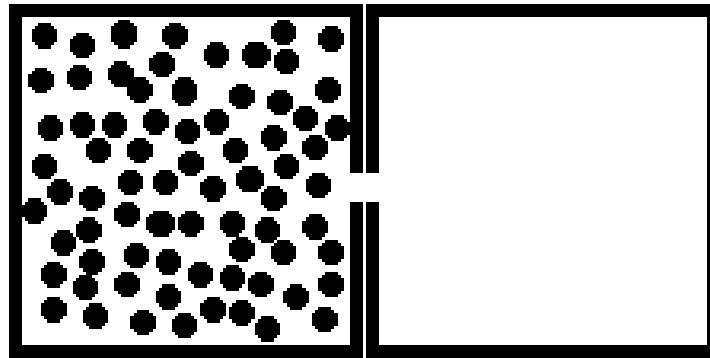
Concentration dependent diffusion coefficient



Statistical averages and stochastic fluctuations

Macroscopic systems towards equilibrium

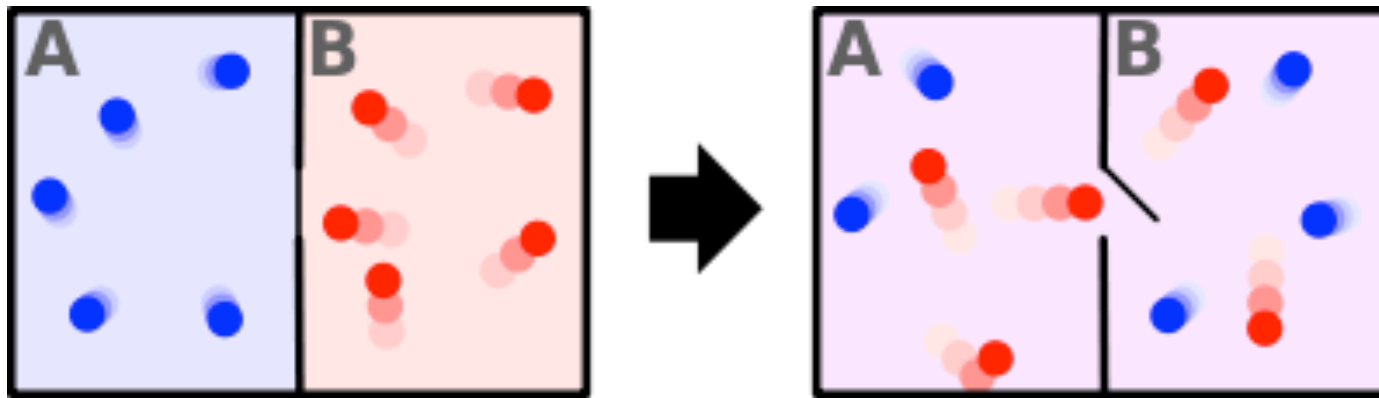
simple example of non-interacting classical particles in a box
(gas diffusion)



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.

$N_{\text{left}}(t)$: number of particles present at time t in the left side
Given $N_{\text{left}}(0)$, what is $N_{\text{left}}(t)$?

Macroscopic systems towards equilibrium



Another version: particles blue/red in both sides
(interdiffusion of two gases):

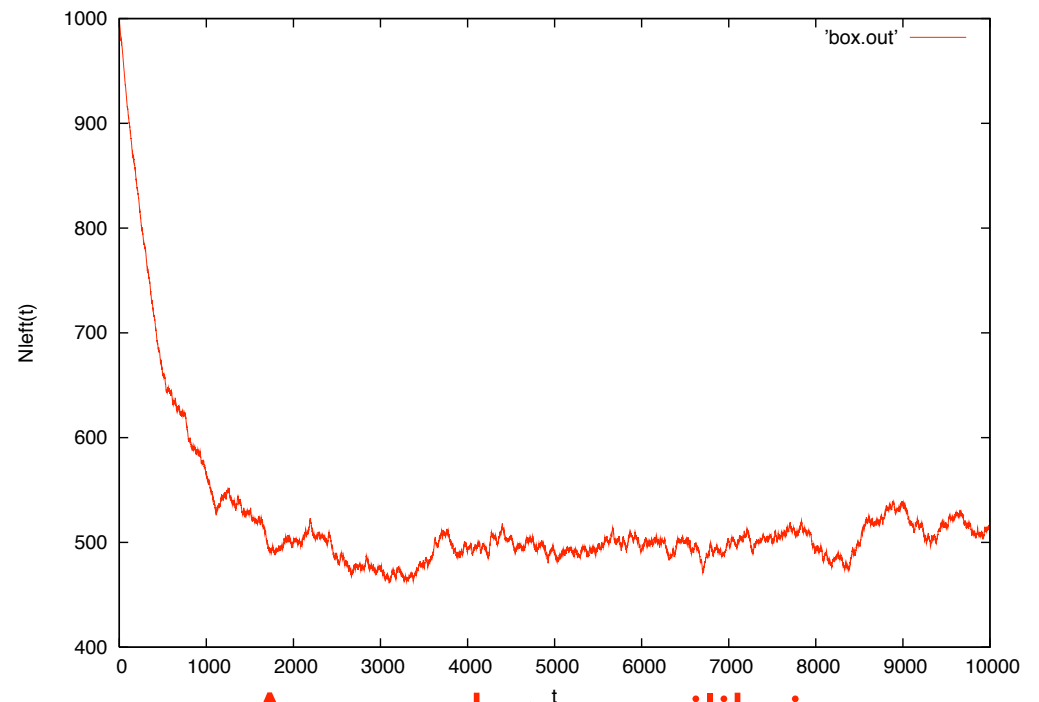
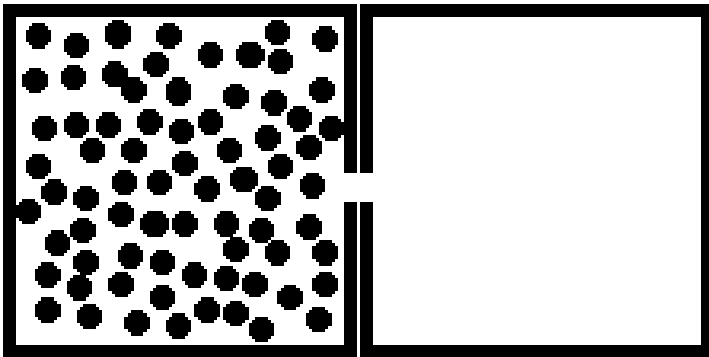
per unit time, one from each side is picked at random and put in the
other side: $N_{\text{left}}^{\text{blue}}(t) + N_{\text{left}}^{\text{red}}(t) = \text{constant}$; $N_{\text{left}}^{\text{red}}(t) = ?$

Paul Ehrenfest (1880-1933)

Stochastic fluctuations

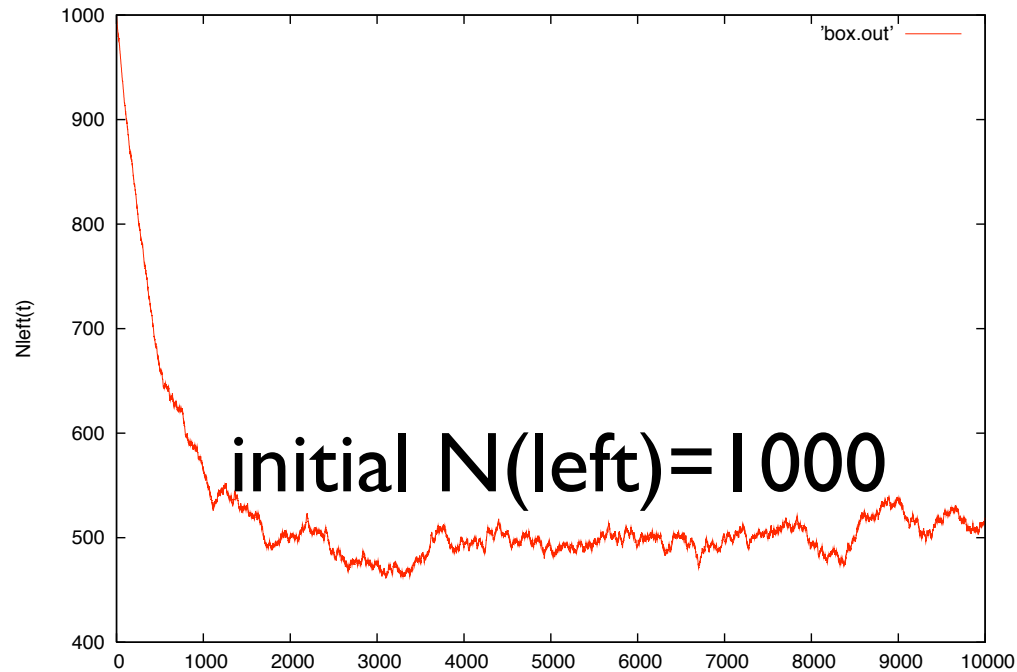
Fluctuations are always present, due to the nature of the system, also when evolving towards equilibrium.

A simple example: non-interacting classical particles in a box (gas diffusion)



Approach to equilibrium
with fluctuations

Stochastic fluctuations



How to reduce **fluctuations**?

- more particles
- average over many simulation runs
- ...

What can we do with fluctuations?

Intrinsic energy fluctuations in the canonical ensemble - I

Remind: $\langle E \rangle = \frac{1}{Z} \sum_s E_s e^{-\beta E_s}$ and $Z = \sum_s e^{-\beta E_s}$, therefore: $\langle E \rangle = -\frac{\partial}{\partial \beta} \ln Z$

Consider the thermal capacity:

$$C_v = \frac{\partial \langle E \rangle}{\partial T} = \dots = -\frac{1}{kT^2} \frac{\partial \langle E \rangle}{\partial \beta}$$

we have

$$\frac{\partial \langle E \rangle}{\partial \beta} = -\frac{1}{Z^2} \frac{\partial Z}{\partial \beta} \sum_s E_s e^{-\beta E_s} - \frac{1}{Z} \sum_s E_s^2 e^{-\beta E_s} = \langle E \rangle^2 - \langle E^2 \rangle = -\langle (\delta E)^2 \rangle$$

where $\delta E \equiv E - \langle E \rangle$

Result:

$$C_v = \frac{\langle (\delta E)^2 \rangle}{k_B T^2}$$

The thermal capacity (or specific heat if considered for each particle) is related to the intrinsic stochastic energy fluctuations

Intrinsic energy fluctuations in the canonical ensemble - II

Since:

$$C_v = \frac{\langle (\delta E)^2 \rangle}{k_B T^2}$$

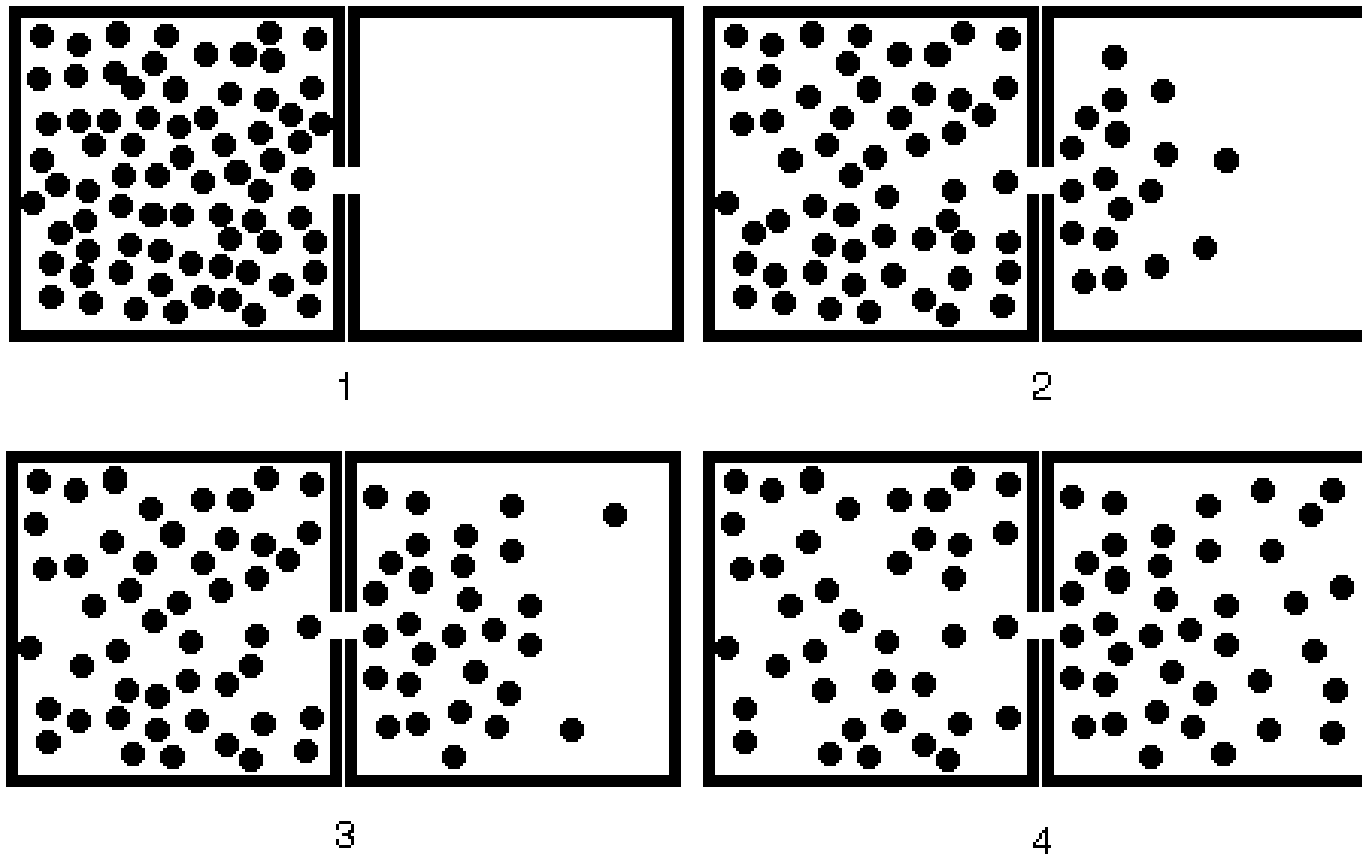
if N is the number of particles, we have:

$$\frac{\sqrt{\langle (\delta E)^2 \rangle}}{\langle E \rangle} = \frac{\sqrt{k_B T^2 C_v}}{\langle E \rangle} \propto \frac{\sqrt{N}}{N} \sim \frac{1}{\sqrt{N}}$$

i.e., the relative energy fluctuations reduce when N is large

(correct; in the thermodynamic limit: $E \rightarrow const.$, macro \sim micro)

Approach to equilibrium



macrostate: specified by the **number** of particles n on the left side;
microstate: specified by the **specific list** of the n particles on the left side

Equilibrium and entropy

$$\text{number of microstates} = \frac{N!}{n!(N-n)!} = \binom{N}{n} = \binom{N}{N-n}$$

The number of microstates for the “particle in a box” model with $N=10$. The macrostate is specified by the number of particles on the left side, n . The total number of microstates for $N=10$ is $2^{10}=1024$

n	# of microstates	$\log(\# \text{ of micr.})$
0	1	0
1	10	2,3
2	45	3,81
3	120	4,79
4	210	5,35
5	252	5,53
6	210	5,35
7	120	4,79
8	45	3,81
9	10	2,3
10	1	0



the most “random”!

Equilibrium =
Maximum number of
possible microstates =
Maximum entropy

(optional)

Entropy: Coincidence method

(S.K. Ma, J. Stat. Phys. 26, 221 (1981))

Equilibrium = Maximum entropy = Maximum number of possible microstates

Too much effort to enumerate all of them!

Alternative procedure (good for computing):

A system evolving in time will duplicate a microstate, before or later...

The longer it takes for duplication, the fewer are the microstates in the corresponding macrostate. Hence, the lower is the entropy.

Idea: measure the ratio of the number of pairs of duplicated microstates to the total number of possible pairs; entropy is the log of the inverse ratio.

E.g.: suppose as in the previous slide $N=10$, and the **macrostate** $n=1$; consider 20 different **microstates** labelled with the “name” of the particle:

8 7 5 10 7 2 4 6 2 10 3 4 3 9 6 5 2 9 2 4

Possible pairs: $20 \cdot (20-1)/2 = 190$. Here: 6 pairs for particle “2”; 1 pair with particle “10” etc etc... Sum all of them: get 15.

Ratio = $15/190$, Entropy: $S \propto \log(190/15) \sim 2.5$

Remind the definition of entropy:

$$S = -k_B \sum_s P_s \ln P_s \quad \text{in the canonical ensemble}$$

$S = k_B \log \Omega$ in the microcanonical ensemble,
where all the microstates
corresponding to a macrostate have
the same energy
(Ω is the number of microstates)

Metropolis method in the canonical ensemble and the simulated annealing

a general purpose global optimization algorithm
(Kirkpatrick S, Gelatt CD Jr, Vecchi MP
Science 220(4598), 671-80, **1983**)

Metropolis and simulated annealing - I

- Stochastic search for global minimum. Monte Carlo optimization.
- The concept is based on the manner in which liquids freeze or metals recrystallize. Sufficiently high starting temperature and slow cooling are important to avoid freezing out in metastable states.

mimics the physical process of annealing by treating the cost function as an “energy” E and sampling the value of E according to the Boltzmann distribution at some artificial temperature T using the Metropolis algorithm

Convergence to the global minimum has been proved for a schedule in which the temperature at the k th iteration $T_k \propto 1/\ln(k)$ and moves are drawn from a Gaussian distribution [8, 9], and also for a schedule where $T_k \propto 1/k$ and moves are drawn from a Cauchy distribution [10].

In practice, a much faster cooling schedule without a convergence proof was used in both the original Kirkpatrick paper, and most applications. In this schedule, $T_k \propto e^{-\lambda k}$ where λ is sometimes adjusted adaptively based on sampling statistics [11] $\Rightarrow T_{k+1} = (1 - \lambda)T_k$. λ is a positive number very close to 0 that controls the cooling speed. The larger λ is, the faster the system cools.

Adaptive schedule are often used

We need:

a cooling schedule, a move generation strategy, and a stopping criterion

8. Geman S, Geman D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions On Pattern Analysis And Machine Intelligence. 1984;6:721–741. [[PubMed](#)] [[Google Scholar](#)]
9. Hajek B. Cooling schedules for optimal annealing. Mathematics of Operations Research. 1988;13:311–329. [[Google Scholar](#)]
10. Szu H, Hartley R. Fast simulated annealing. Physics Letters A. 1987;122:157–162. [[Google Scholar](#)]

Metropolis and simulated annealing - II

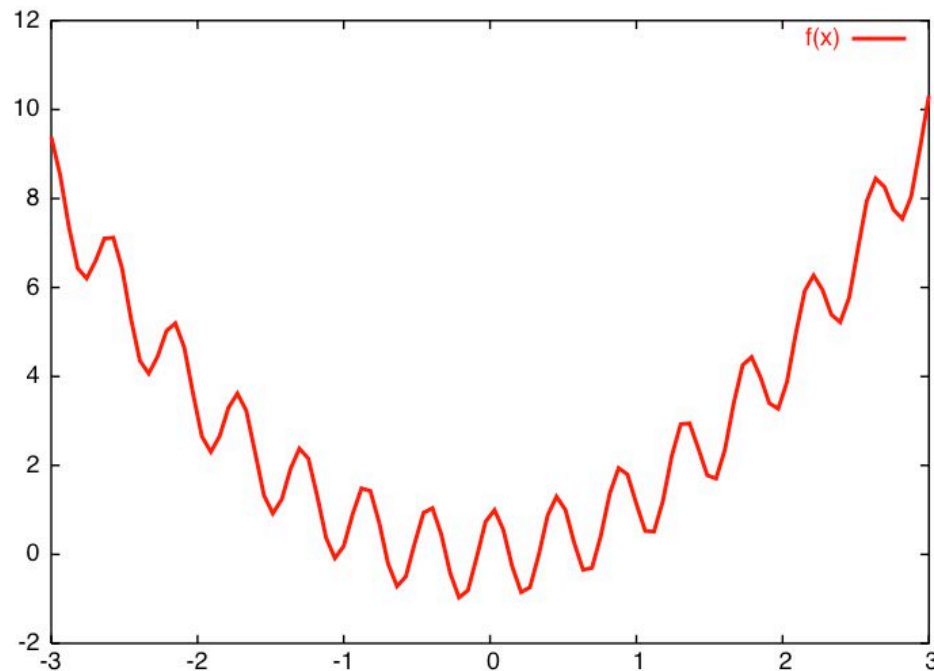
usual
Metropolis
procedure
in the
canonical
ensemble

- Thermodynamic system at temperature T , energy E .
- *Perturb configuration (generate a new one).*
- *Compute change in energy dE . If dE is negative the new configuration is accepted. If dE is positive it is accepted with a probability given by the Boltzmann factor : $\exp(-dE/kT)$.*
- *The process is repeated many times for good sampling of configuration space.*
- **then the temperature is slightly lowered and the entire procedure repeated, and so on, until a frozen state is achieved at $T = 0$.**

Example

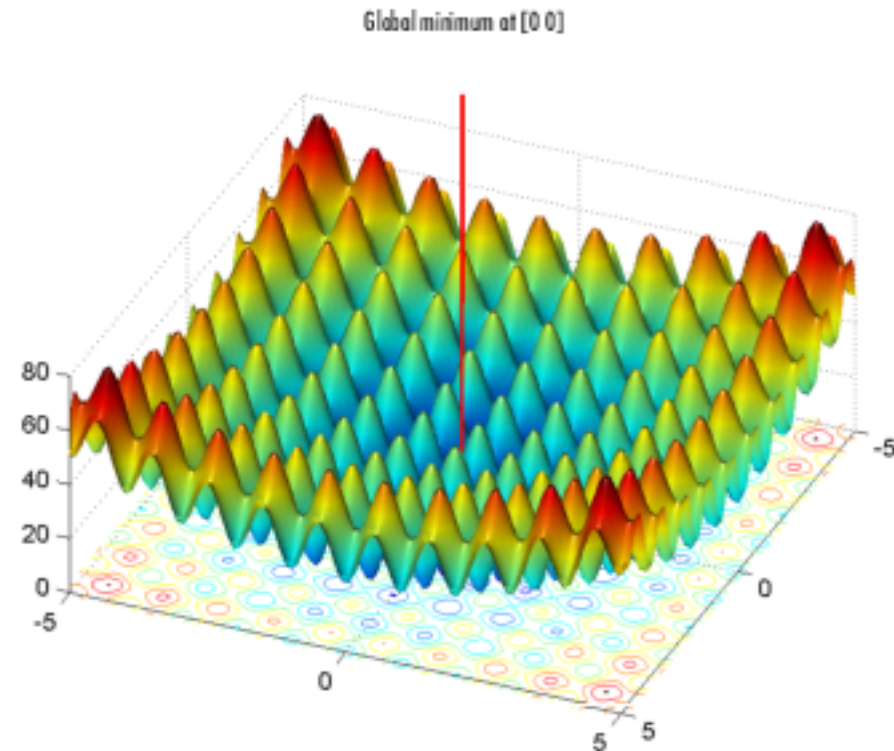
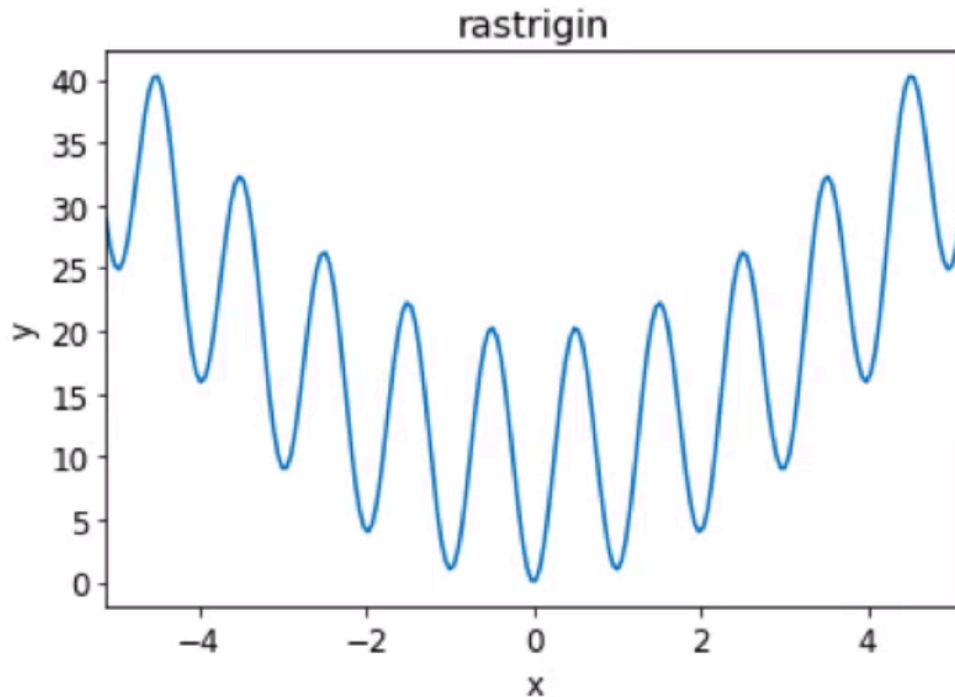
in **simulated_annealing.f90**:
minimization of

$f(x) = (x+0.2) * x + \cos(14.5 * x - 0.3)$
considered as an energy function and
using a fictitious temperature



Rastrigin function:

- non-convex *function* used as a performance test problem for optimization algorithm
- typical example of non-linear multimodal *function*;
- first proposed by *Rastrigin* as a 2-dimensional *function*; later generalized by Rudolph.



$$f(\mathbf{x}) = nA + \sum_{i=1}^n [x_i^2 - A \cos(2\pi x_i)]$$

Function to be minimized: $f(\mathbf{x})$; Starting point: \mathbf{x} , $\mathbf{fx}=f(\mathbf{x})$

	initial (high) temperature:	temp
Annealing schedule:	annealing temperature reduction factor:	tfactor (<1)
	number of steps per block:	nsteps
'ad hoc' parameter for trial move:	scale	

```
DO WHILE (temp > 1E-5) ! anneal cycle
```

```
DO istep = 1, nsteps
```

```
CALL RANDOM_NUMBER(rand) ! generate 2 random numbers; dimension(2) :: rand
```

```
x_new = x + scale*SQRT(temp)*(rand(1) - 0.5) ! stochastic move
```

```
fx_new = func(x_new) ! new object function value
```

```
IF (EXP(-(fx_new - fx)/temp) > rand(2)) THEN ! success, save
```

```
fx = fx_new
```

```
x = x_new
```

```
END IF
```

```
IF (fx < fx_min) THEN
```

```
fx_min = fx
```

```
x_min = x
```

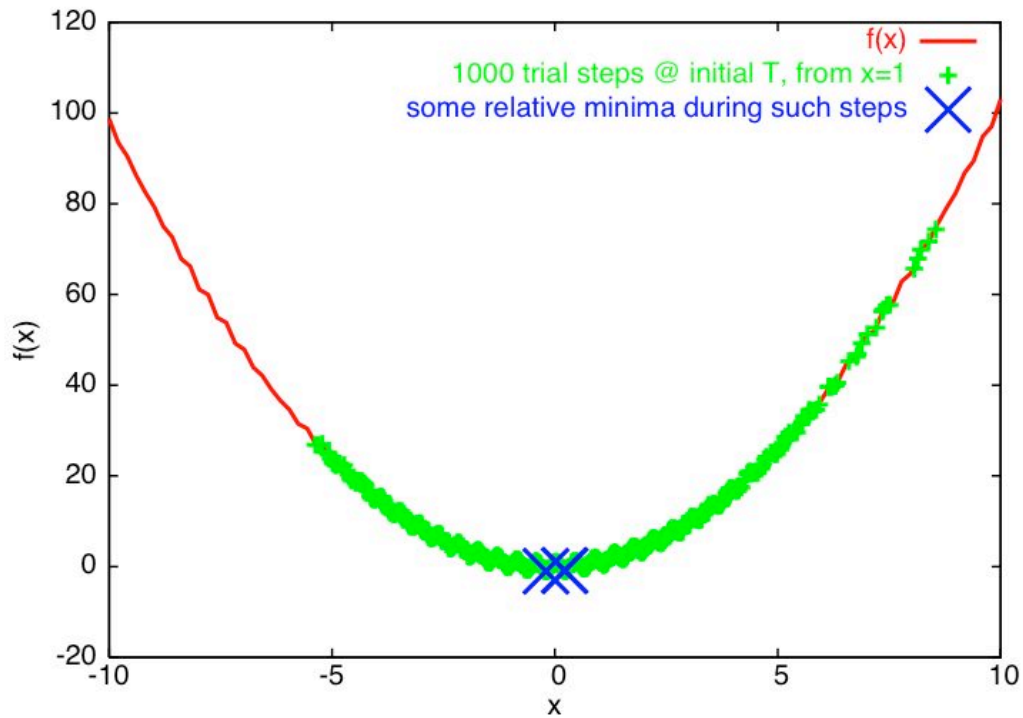
```
PRINT '(3ES13.5)', temp, x_min, fx_min
```

```
END IF
```

```
END DO
```

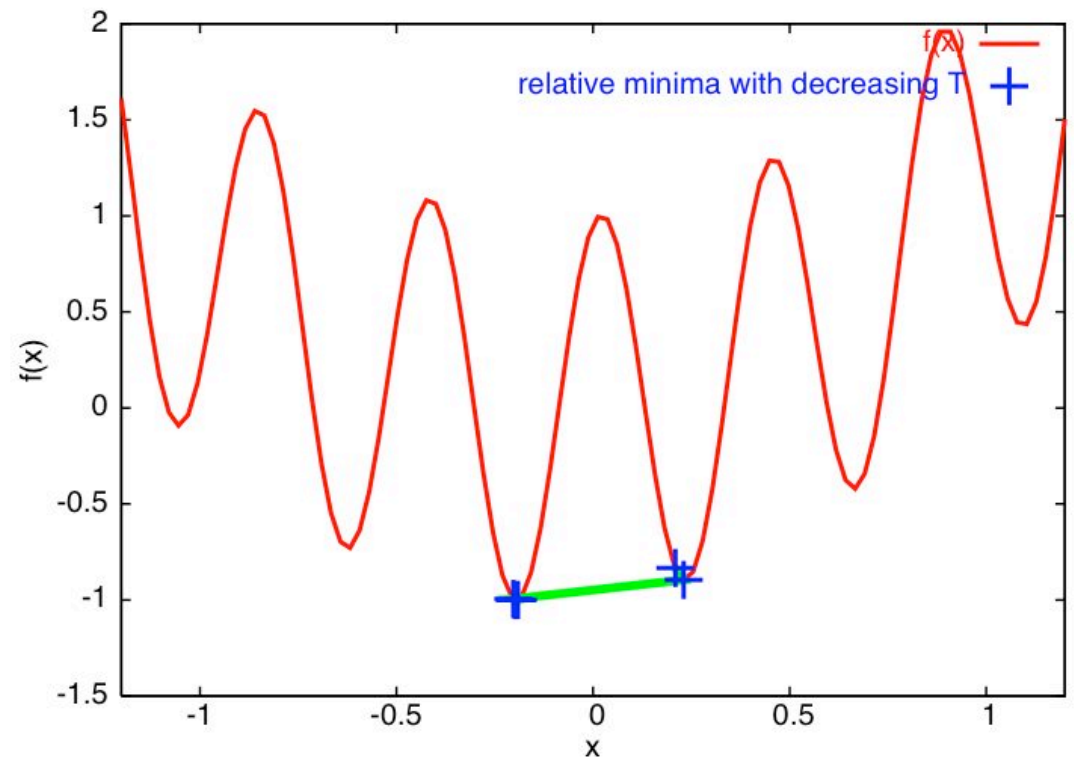
```
temp = temp * tfactor ! decrease temperature
```

```
END DO
```



initial T : 10 (K_B units)
 initial x : 1.000000
 initial $f(x)$: 1.137208

final T : 2.50315E-01
 final x : -1.95067E-01
 final $f(x)$: -1.00088E+00





**993SM - Laboratory of
Computational Physics
Lab session 10 - part 1
May 15, 2020**

Maria Peressi



**993SM - Laboratory of
Computational Physics
Lab session 10 - part 2
May 15, 2020**

Maria Peressi



**993SM - Laboratory of
Computational Physics
Lab session 10 - part 3
May 15, 2020**

Maria Peressi



**993SM - Laboratory of
Computational Physics
Lab session 10 - part 4
May 15, 2020**

Maria Peressi