Simulations in Statistical Physics

Course for MSc physics students

Janos Török

Department of Theoretical Physics

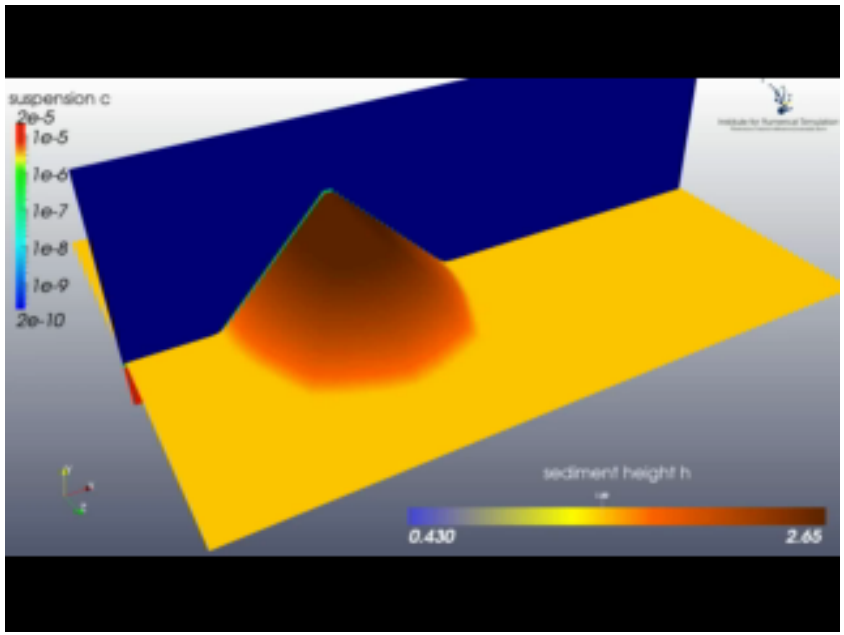
September 15, 2015

Information

- Coordinates:
  - Török János
  - Email: `torok@phy.bme.hu`, `torok72@gmail.com`
  - Consultation:
    - F III building, first floor 6 (after the first stairs to the right, at the end of the corridor), Department of Theoretical Physics
    - (Department door is open if light to the right is green push the door HARD!)
    - Upon demand (Email)
- Webpage: `http://dtp.physics.bme.hu/`
- Homework: `http://newton.phy.bme.hu/moodle`

Required knowledge

- Knowledge of basic statistical physics
- C, or C++ language (only basic things)
- English



Examination requirements

- Signature
  - Mid November: home work
  - A problem to be solved by simulation
  - Code written in C or C++, which compiles easily
  - Documented working code (no extra libaries except for `gsl`)
  - Using fancy visualization techniques does not impove the mark which is given for the algorithm, the efficiency of the code and the solution of the problem
  - A pdf documentation of the results and explanation (3-5 pages)
  - Language: English, Hungarian
- Exam: mark
  - 3/5: From the code and documentations
  - 2/5: Lecture material
    - Both must be at least 2 to have a final note larger than 1
  - Presentation random part of the code
  - Language: English, Hungarian, German, French

Literature

- D.W. Heermann: Computer simulation methods in theoretical physics, Springer, 1995
- D. Landau and K. Binder: A guide to Monte Carlo simulations in statistical physics (Cambridge UP, 2000)
- D. Rapaport: The art of molecular dynamics programming (Cambridge UP, 2004)
- J. Kertész and I. Kondor (eds): Advances in computer simulation (Springer, 1998)
- W.G. Hoover: Molecular Dynamics (Springer, 1986)

Overview of statistical physics

Aim

- Microscopic explanation of thermo dynamics
- Calculate macroscopic properties from microscopic principles
  - Here simulate
- Explain phenomena (phase transitions, pattern formation, etc.)

Major parts

- Equilibrium
- Non-equilibrium
  - Perturbation of an equilibrium system
  - Far-from equilibrium system

Definitions in statistical physics

- Isolated system: No interactions with the world
- Closed system: Only energy transfer with the world
- Reservoir: Part of an isolated or closed system which is much larger than the rest and any change in the rest leaves this part unaffected
- Microstate: a point in the phase space, snapshot of the system with all required quantities (e.g. position, speed, etc.)
- Macrostate: thermodynamic or hydrodynamic state.
- Equilibrium: No flow of energy in the system
- Detailed balance: in thermodynamic equilibrium

$$\pi_i P_{ij} = \pi_j P_{ji}$$

π_i: probability of state i, P_{ij}: transition probability $i \to j$

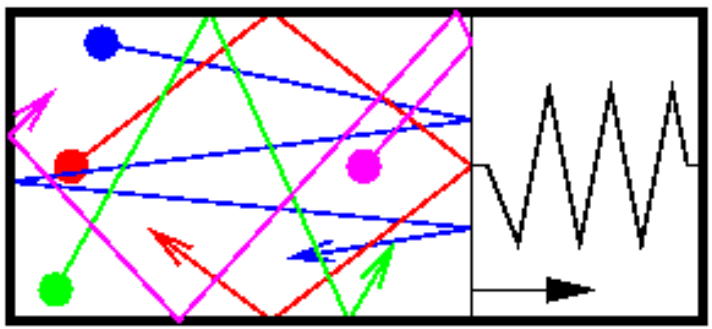
Averages

- Time average:

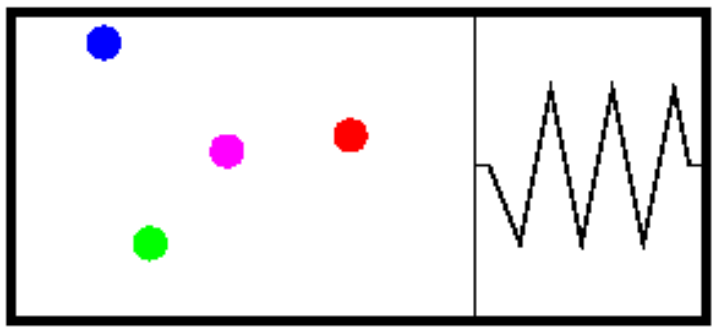
$$\overline{A} = \lim_{T\to\infty} \frac{1}{T} \int_0^T A(q(t), p(t))dt$$

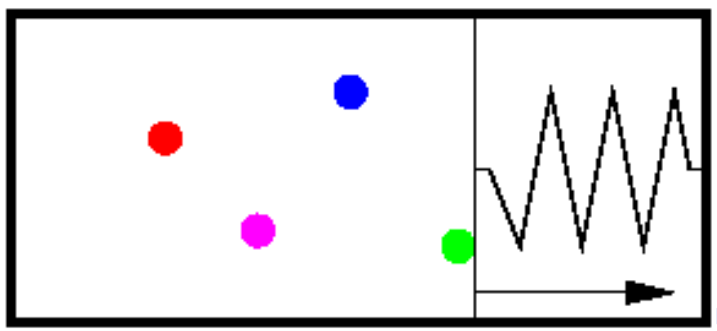
- Ensemble average:

$$\langle A \rangle = \frac{1}{h^{3N}(N!)} \int A(q,p) P^{eq}(q,p) dqdp$$



F





F

Averages

- Time average:

$$\overline{A} = \lim_{T\to\infty} \frac{1}{T} \int_0^T A(q(t), p(t))dt$$

- Ensemble average:

$$\langle A \rangle = \frac{1}{h^{3N}(N!)} \int A(q,p) P^{eq}(q,p) dqdp$$

  E.g. $P^{eq}(q,p) = \exp(-\beta H)$.

- Equivalence: Ergodicity, Thermodynamic limit $N \to \infty$, $T \to \infty$.
- Problems:
  - Order of limits (glasses)
  - Non-equilibrium: $T \to \infty$

Fluctuation-dissipation theorem

- Dynamical system $H_0(x)$ subject to thermal fluctuations
- Observable $x(t)$ fluctuates around $\langle x \rangle_0$.
- Power spectrum of fluctuations of x: $S_x(\omega) = \hat{x}(\omega)\hat{x}^*(\omega)$
- Linear perturbation of the Hamiltonian: $H(x) = H_0(x) + fx$
- Susceptibility (linear response):

$$\langle x(t) \rangle = \langle x \rangle_0 + \int_{-\infty}^{t} f(\tau)\chi(t-\tau)d\tau$$

- The Fluctuation-dissipation theorem relates the above as

$$S_x(\omega) = \frac{2k_B T}{\omega} \mathrm{Im}\hat{\chi}(\omega)$$

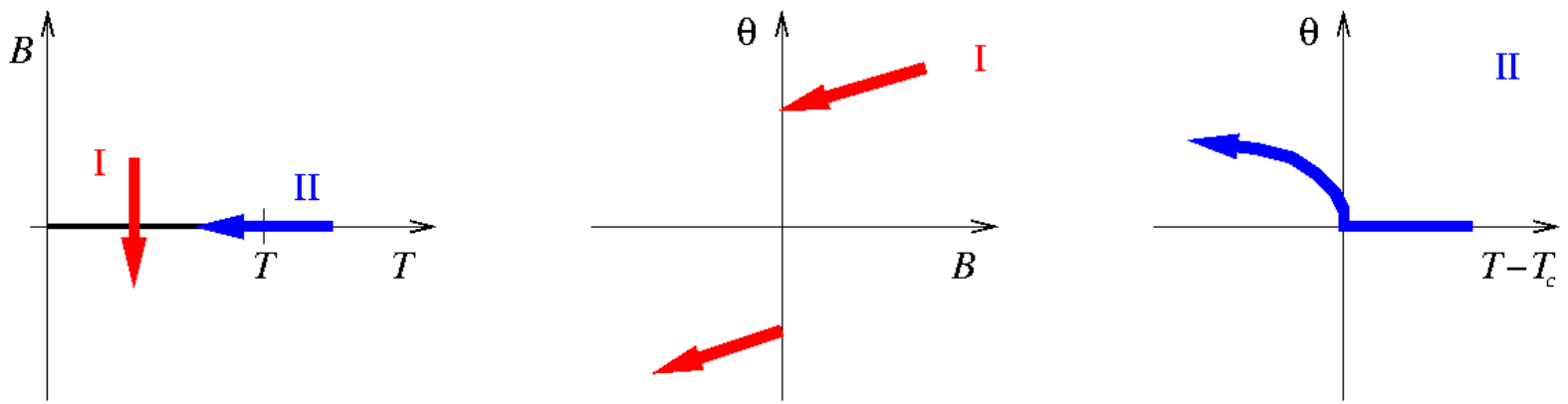
- Can be used to define temperature

Phase transitions

Equilibrium	order parameter	Non-equilibrium	order parameter
liquid-gas	density	traffic jam	flux
ferromagnetic	magnetization	flocking	average speed
		jamming	$\phi_c - \phi$
		glass	replicas

- Order of phase transition: which derivative of Gibbs free energy becomes discontinuous
- Better classification:
  - First order: discontinuous transition (latent heat)
  - Second order: continuous transition, order parameter is continuous but susceptibility is divergent

Phase transitions



- Order of phase transition: which derivative of Gibbs free energy becomes discontinuous
- Better classification:
  - First order: discontinuous transition (latent heat)
  - Second order: continuous transition, order parameter is continuous but susceptibility is divergent

Correlation function

- e.g. Magnetic systems

$$G(r) = \langle s(R)s(R+r)\rangle - \langle s\rangle^2$$

- Close to the critical point:

$$G(r) = r^{-(d-2+\eta)} \exp(-r/\xi),$$

where

$$\xi \propto |T - T_c|^{-\nu}$$

is the correlation length. The correlation length, i.e., the characteristic size of the regions, where the fluctuations are correlated diverges at the critical point.

- ν and η are critical exponents.

Correlation function

- Near to the critical point G is a generalized homogeneous function of its variables:

$$G(r, t, h) \propto b^{-2\beta/\nu} G(r/b, b^{y_t} t, b^{y_h} h),$$

  where $t = (T - T_c)/T_c$ and $t \to 0$, $h \to 0$.
- The susceptibility

$$\chi = \beta V \int G(\mathbf{r}) dr^3 = \beta \langle (s - \langle s \rangle)^2 \rangle$$

- Magnetization (OP), susceptibility, specific heat

$$\chi = \frac{\partial M}{\partial h}, \qquad M = \frac{\partial F}{\partial h}, \qquad C = \frac{\partial F}{\partial T}$$

Scaling relations

- $C(h=0) \propto |t|^{-\alpha}$
- $M(h=0) \propto (-t)-\beta,\ t<0$
- $\chi(h=0) \propto |t|^{-\gamma}$
- $M(t=0) \propto h^{1/\delta}$
- 8 exponents: $\alpha, \beta, \gamma, \delta, \eta, \nu, y_t, y_h$
- Scaling relations (d dimension):
  - $y_t = 1/\nu,\ y_h = d - \beta/\nu$
  - $\alpha + 2\beta + \gamma = 2$
  - $\delta = 1 + \gamma/\beta$
  - $d\nu = 2 - \alpha$
  - $\nu = \gamma/(2-\eta)$
- Two independent exponents left $\rightarrow$ universality classes

Simulations

Experiments	*Simulations*
Principle of measurement	Algorithm
Apparatus	Program + Hardware
Calibration	Calibration + Debugging
Sample	Sample
Measurement	Run

Data collection

Analysis

Simulations

Experiments	*Simulations*
Principle of measurement	Algorithm
Apparatus	Program + Hardware
Calibration	Calibration + Debugging
Sample	Sample
Measurement	Run

Data collection

Analysis

Marked ones: Computer codes!

Programming languages

Simulations codes

- System size must be large
  - Phase transition $\xi \to \infty$
  - Real systems $N \sim 10^{23}$ (memory $< 10^{11}$)
- Simulation time should be long
  - Relaxation time
  - Interesting phenomena take long
  - Separation of time scales

Must be efficient!

It is not bad if program is readable and extensible...

Sample preparation

- Sometimes it is also a simulation

Data analysis

- Anything may happen

Programming languages

Problem to solve:

- Fill an array with product of two random numbers
- Calculate the average of them

python

```python
import random
random.seed(12345);
N = 10000
s = []
for i in range(0,N):
  s.append( random.random() * random.random() )

av = 0
for i in range(0,N):
  av += s[i]

print av/N
```

matlab

```matlab
N = 10000;
s = zeros(N,1);
rng( 12345 );
for i = 0:N
  s(i) = rand * rand;
end
% s = rand(N,1);
av = 0;
for i = 0:N
  av = av + s(i);
end
av = av / N;
% av = sum( s ) / N;
disp( av );
```

Programming languages

```
N = 10000;
s = zeros(N,1);
rng( 12345 );
for i = 0:N
  s(i) = rand * rand;
end
% s = rand(N,1);
av = 0;
for i = 0:N
  av = av + s(i);
end
av = av / N;
% av = sum( s ) / N;
disp( av );
```

```
import random
random.seed(12345);
N = 10000
s = []
for i in range(0,N):
  s.append( random.random() * random.random() )

av = 0
for i in range(0,N):
  av += s[i]

print av/N
```

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int main(int argn,char * argv[])
{
  int i,N;
  double *s;
  double av, rm1;

  N=10000000;
  s = (double *)calloc(N, sizeof(double));
  srand(12345);
  rm1 = 1.0 / RAND_MAX;

  for (i=0; i<N; i++) {
/*    s[i] = (double) rand() * rm1 * rand() * rm1;*/
    s[i] = (double) rand() * rand() / RAND_MAX / RAND_MAX;
  }
  av = 0.0;
  for (i=0; i<N; i++) {
    av += s[i];
  }
  printf("%lg\n", av / N);
}
```

Optimization

- Multiplication vs. Division (*not so* old computers)

```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

int main(int argn,char * argv[])
{
  int i,N;
  double *s;
  double av, rm1;

  N=10000000;
  s = (double *)calloc(N, sizeof(double));
  srand(12345);
  rm1 = 1.0 / RAND_MAX;

  for (i=0; i<N; i++) {
/*    s[i] = (double) rand() * rm1 * rand() * rm1;*/
    s[i] = (double) rand() * rand() / RAND_MAX / RAND_MAX;
  }
  av = 0.0;
  for (i=0; i<N; i++) {
    av += s[i];
  }
  printf("%lg\n", av / N);
}
```

Optimization

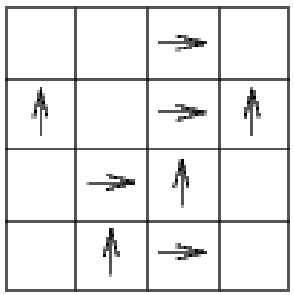
- Programming language
  - In example C is 20 times faster than python
  - On old computers with multiplication is 20% faster
  - Matlab, Maple, Mathematica are expensive
  - Clusters have C, and C++
- Optimization
  - Parallelization
  - Indexing Careful usage of pointers
  - Reformulate operations
  - Does not always worth the pain
  - `gprof`

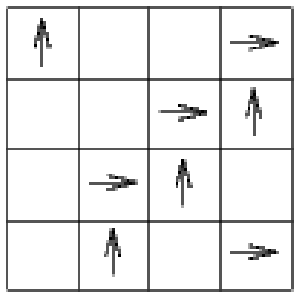
gprof

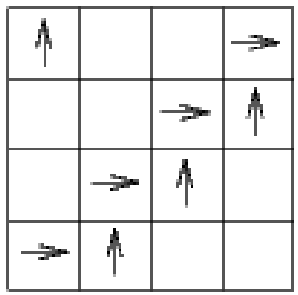
```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 37.66     56.83    56.83 3248064862     0.00     0.00  is_in_community
 25.99     96.05    39.22  1000000     0.04     0.04  e_erode
 11.55    113.47    17.43 21355853     0.00     0.00  weighted_random_link
  6.33    123.03     9.55 11078805     0.00     0.00  weighted_random_link_ban_list
  3.02    127.58     4.55  8406648     0.00     0.01  e_info
  2.77    131.75     4.18                             main
  2.26    135.16     3.40 197988614     0.00     0.00  ct_weight
  2.10    138.33     3.17        4   792.50   792.50  clear_data
  1.85    141.12     2.79 12949626     0.00     0.00  e_single
  1.73    143.74     2.62 164260875     0.00     0.00  ranksz
  1.60    146.16     2.42 12774907     0.00     0.00  strengthen
  0.97    147.62     1.46 19359356     0.00     0.01  communicate
  0.88    148.94     1.33 248428917     0.00     0.00  is_internet
  0.32    149.43     0.48    15380     0.03     0.03  random_agent_with_group_sex
  0.31    149.90     0.47  2042439     0.00     0.00  e_share
  0.24    150.25     0.36                             seed3
```

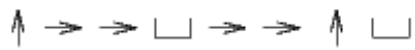
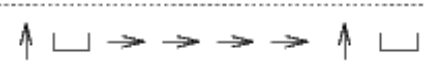
Optimization

- Programming language
- Optimization
  - Careful with time
  - Too much optimization prevents further development





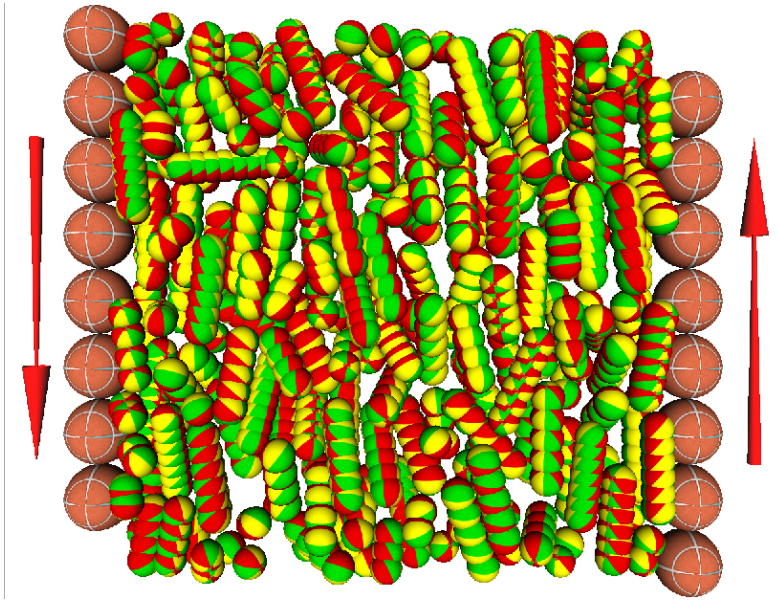


1)	**Jobbra haladó** : 01101100	
2)	**Felfelé haladó** : 10000010	
3)	Shift 1) : 00110110	
4)	2) AND 3) : 00000010	
5)	3) ADC 4) : 00111000	
6)	5) AND NOT 3) : 00001000	
7)	3) - 4) + 6) : 00111100	

Optimization

- Programming language
- Optimization
  - Careful with time
  - Too much optimization prevents further development
  - Optimize only working code!
- Algorithm
  - The war can be won here

Simulations



- Do what nature does
  - Molecular dynamics
  - Hydrodynamics
- Make use of statistical physics
  - Monte-Carlo dynamics
  - Simulate simplified models
  - Much smaller codes!