Simulation Methods in Statistical Physics

FK7029
2015
①

contact: - Astrid de Wijn <dewijn@fysik.su.se>
office (4:3023)
- Jesper Norell <jespernorell@gmail.com>
office (4:3059)
- course website http://www.syonav.net/FK7029

Give us your email address.

Book: Frenkel & Smit Understanding molecular Simulation
available electronically from the library
Other book: Tuckerman, Statistical Mechanics: Theory & Mol. Sim

Prerequisites: - some stat phys (Supriya's course recommended)
- some basic programming.

Compulsory for Computational Physics Master

Course consists of
- lectures
- few lab hours
- Jesper's office hours
- programming
- hand-in exercises } examination
- written exam

Examination: programming - theory 50-50 pass both

Lectures:
- Schedule on course website
- Reading material as well
- will scan lecture notes

Lab hours:
- Get you started
- lecture/lab depends on how fast I go. } bring laptops

Office hours (somewhere in C4 corridor)
- Sit in a room and work: program/exercises/read, with Jesper there
- Strongly recommended
- When?

Programming:
- many small projects, make up 2 big programs.
- one every lecture.
- small report
- pass/fail + overall grade late = fail
- deposit only once per project, 3 times in total

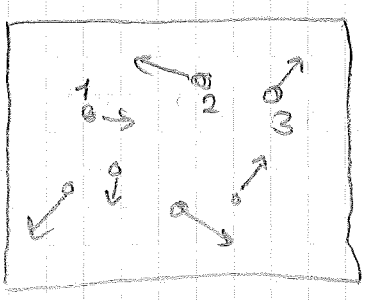
Theory:
- deadlines hand-in on website, late = fail, no doubles } 60% each = 120%
- 4 points per exercise
- exam

Today: - introduction
- some stuff to help you get started with prog

What's the problem?

statistical physics ≡ many particles.

$\vec{r}_1, \vec{v}_1, \vec{r}_2, \vec{v}_2, \vec{r}_3, \vec{v}_3 \ldots \ldots \vec{r}_N, \vec{v}_N$



in general, particles interact: pot. energy
- $V = \sum_{i,j} V_0(\vec{r}_i - \vec{r}_j)$
- or more complicated $V(\vec{r}_1, \ldots \vec{r}_N)$

example gas in a box or lecture room
$N \sim N_A \approx 6 \cdot 10^{23}$

Equations of motion (classical)

$\dot{\vec{r}}_1 = \vec{v}_1 \quad \dot{\vec{r}}_2 = \vec{v}_2 \quad \dot{\vec{r}}_3 = \vec{v}_3$

$\dot{\vec{v}}_1 = -\frac{1}{m_1} \frac{\partial V(\vec{r}_1 \ldots \vec{r}_N)}{\partial \vec{r}_1}$

$\dot{\vec{v}}_2 = -\frac{1}{m_2} \frac{\partial V(\ldots)}{\partial \vec{r}_2}$

6N coupled, nonlinear differential equations

⇒ too difficult / too much work to solve with pen and paper.

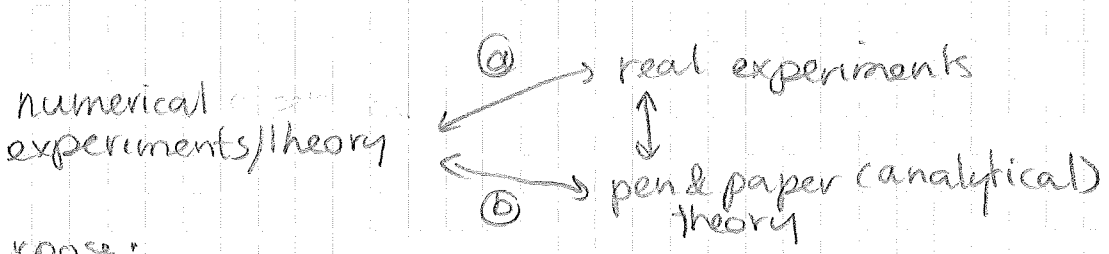
not interested in individual particles but in <u>global averages</u>
(pressure, density profile, current, etc...)

Equilibrium Statistical Physics is a powerful general theory for N large.
- complex interactions make it hard to use
- does not apply in systems out of equilibrium
  ⇒ some ad-hoc approaches still exist, but cumbersome and very specific (f.e. kinetic theory of gases)

air in this room: not in equilibrium. (we move and breathe, pressure outside changes, temperature, humidity)

What to do when pen & paper won't do?
[not enough time (too lazy), or too stupid]

=> simulate it (computer does the work!)



Purpose:
ⓐ quantitative, realistic simulations, compare to experiments
ⓑ simple model systems just beyond theoretical understanding to support development of analytical theory

in simulations you can switch stuff on & off and see everything
(we cannot really simulate N_A particles of course)

Types of simulations, most are:
- Molecular Dynamics (MD) or
- Monte Carlo (MC)
(one programming project each)

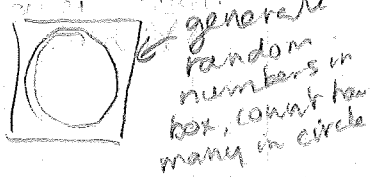
MD: solve the classical time-dependent, nonlinear equations of motion explicitely (of molecules or something)

example gas in box

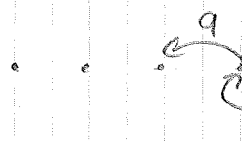
$\gamma = (r_1, r_2, r_3 \ldots r_N, v_1, v_2, v_3 \ldots v_N)$ phase space variable } → dynamical system

eq. o. m $\dot{\gamma} = f(\gamma)$

MC: some kind of random sampling
→ to replace dynamics (usually based on stat. phys theory)

Other example calculate π


example random walk on a line



What does the probability distribution look like after a while

- take one walker draw random steps, let it run around
- repeat many times
- not exact solution but reasonable approximation

Truth: line between MD & MC is vague

Structure of a simulation

step 1 initialise

- declare variables double *r double *v & set parameters
- allocate memory, malloc(3N*sizeof(double))
- set initial conditions
  $r_1 = \ldots$, $r_2 = \ldots$
  $v_1 = \ldots$, $v_2 = \ldots$

Step2 Run integrate eq. of motion or MC algorithm in a loop

Step3 Analyse (on the fly or at the end) calculate interesting quantities.

Step4 Output (on the fly or at the end)

...

+ additional fiddling of course

There are some big codes that take care of a lot of the details for you, such as LAMMPS, Gromacs

---

Practical stuff about programming

- bugs (most time spent on finding them)
- available resources (cpu & memory)

tips for bugs and related hassle

- look at tips on website
- think before you type
- keep it simple! programmer time vs computer time
- use subroutines & functions
- use descriptive variable names
- write comments
- indent the code
- use a smart editor (folds, syntax highlighting)
- use debugging tools such as gdb, valgrind, cachegrind.
- KEEP THE OVERVIEW

- If it doesn't work, some part is not doing what you think. You must find other ways to check your code.

Show folds in vim & indentation in dipole code

Scientific considerations

- model ≠ real system
- accuracy ⟷ resources

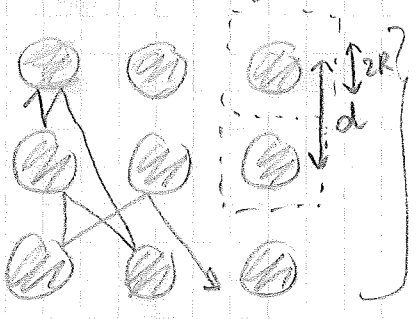
⇒ model considerations
- basic interactions
- what quantities to keep constant (P, V, N, T, E)
- external forces
- boundary conditions
- integration algorithms
- target quantities
- particle number
- initial conditions
- time scales

Simple example of some of these issues

Particle bouncing between other particles?
(i.e asymmetric gas mixture, classical picture of electron in solid)

Lorentz gas: point particle + fixed array of spherical scatterers

regular L-gas.



= model (in 2d here)

pinball!

Let's say we do MD.

Initialise (step 1)
variables: $\vec{r}$, $\vec{v}$, t?
parameters v, R, d,
initial conditions?

→ which ones should you avoid?

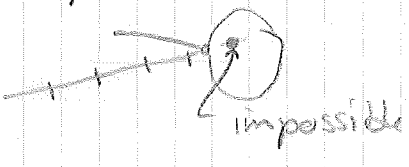
gives scatterer edge positions (do not want to store 10^23 scatterers)

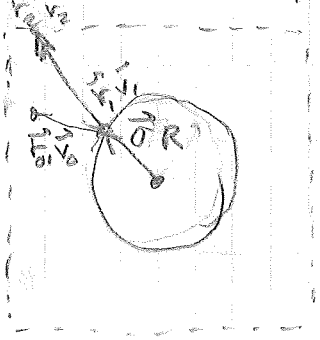
step 2: Run (MD) sketch integration algorithm

time increments? $t \to t+\Delta t$ $(x,y) \to (x+v_x\Delta t, y+v_y\Delta t)$ (1)

Not good

⇒ calculate intersections to scatterer or boundary or area around scatterer





Sinai billiard
- single circular scatterer
- periodic boundaries

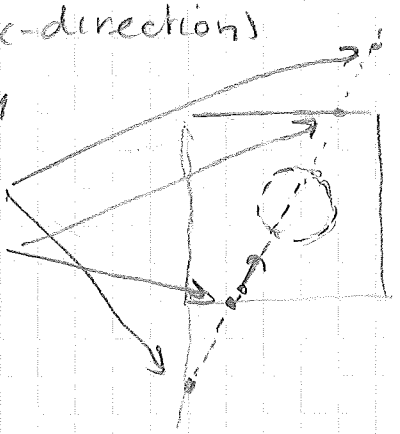
(assume scatterer is at (0,0))

boundaries: $x + v_x \Delta t = \pm \frac{d}{2}$ (x-direction)

$y + v_y \Delta t = \pm \frac{d}{2}$

$\Delta t = \frac{1}{v_x}\left(\pm \frac{d}{2} - x\right)$

$\Delta t = \frac{1}{v_y}\left(\pm \frac{d}{2} - y\right)$.



scatterer collision.

$$(x + v_x \Delta t)^2 + (y + v_y \Delta t)^2 = R^2$$

$$\Rightarrow \Delta t^2 v^2 + \underbrace{2(x v_x + y v_y)}_{b} \Delta t + x^2 + y^2 = R^2$$

$\Rightarrow$ collision if $b^2 - 4v^2(x^2+y^2) > 0$

$$\Delta t = \frac{1}{2v^2}\left[-b \pm \sqrt{b^2 - 4v^2(x^2+y^2)}\right]$$

So, max 6 possible Δt.

earliest one that is in the future is correct.

update now with (1), and if collision, reflect $\vec{v}$ according to

$(v_x, v_y) \rightarrow (1-2\sigma)(v_x, v_y)$ $\quad \vec{\sigma} = \frac{1}{R}(x, y)$

then put back in $[-d/2, d/2]\,[-\frac{d}{2}, \frac{d}{2}]$

produces sequence of boundary crossings and collisions

Question: how best to implement this, (minimising cpu time)

Everything you do cost cpu effort!

function	time needed	
drand48()	1	beware of these
*, +	~1/10	lots of +, * etc to calculate these
sin	5	(though well optimised, still expensive)
cos	5	
sincos	5	← use this if you need sin & cos
log	5	
exp	4	
sqrt	2	

initialisation happens once
update algorithm gets used over and over.

make sure only do calculate sqrt in scatterer collision time when needed and only once!

use cachegrind to see what cpu spends much time on

end lecture 1