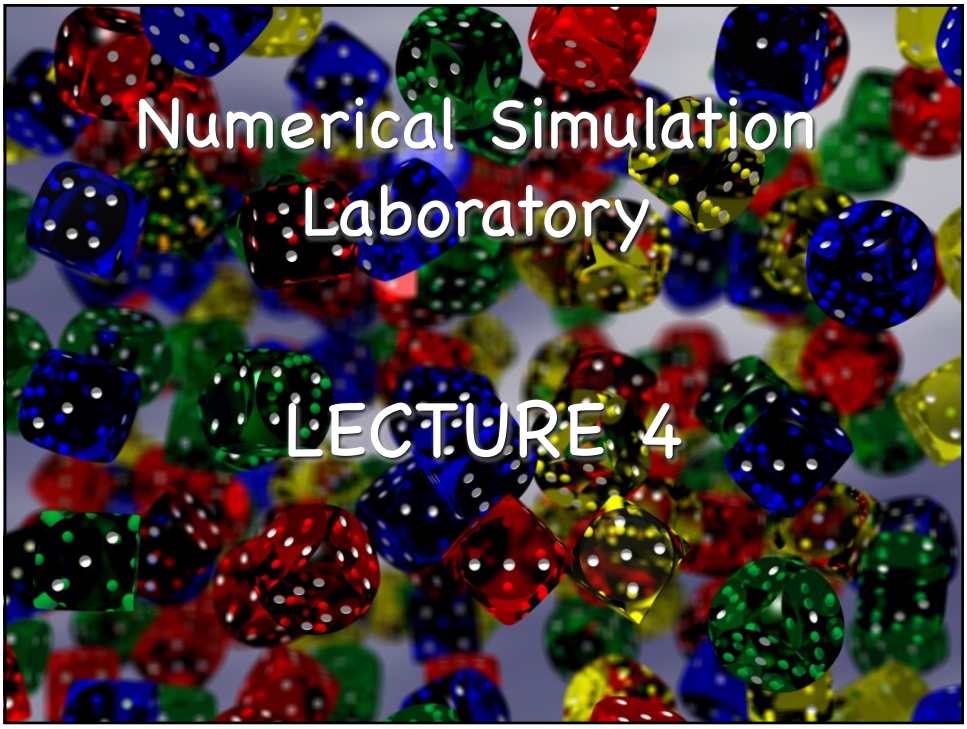


1

Outline

- Statistical Mechanics an introduction
- Microcanonical ensemble and the ergodic hypothesis
- Computer simulations
- Introduction to Molecular Dynamics
- Chaotic dynamics
- Integration algorithms
- Studing small systems: periodic boundary conditions
- Potential energy: minimum image-convention & cut-off
- Examples: LJ fluid
  - Initialization
  - Reduced units
  - Tail corrections
  - Temperature & pressure
- MD code

2

2

Introduction

- Thermodynamics began with the observation that matter can exist in macroscopic states which are stable and do not change in time (unless some external influence acts to change it)

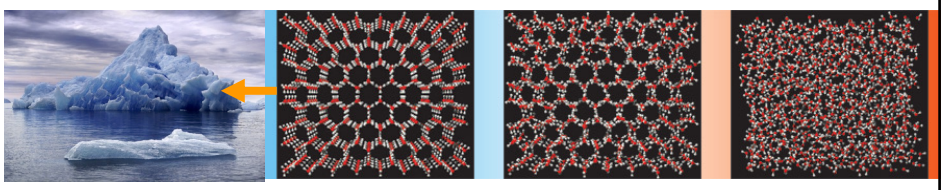
- These equilibrium states are characterized by definite mechanical properties
- The amazing feature of macroscopic systems is that, even though they contain many degrees of freedom ($\approx 10^{23}$) in chaotic motion, their thermodynamic state can be specified completely in terms of a few parameters, called state variables. In general, there are many state variables which can be used to specify the thermodynamic state of a system, but only a few are independent.
- In practice, one chooses state variables which are accessible to experiment and obtains relations between them. Then, the "machinery" of thermodynamics enables one to obtain the values of any other state variables of interest

3

3

Statistical mechanics

- If particles (atoms, molecules, ...) obey certain microscopic laws with specified inter-particle interactions, what are the observable macroscopic properties of a system containing a very large number of such particles?
- The goal of statistical mechanics is to predict the equilibrium macroscopic properties of bodies on the basis of their microscopic structure/laws (as governed by Schrödinger's equation or Newton's laws of motion)



Temperatura

- Gradi di libertà in un sistema macroscopico: Numero di Avogadro

$$N_A \cong 6.02 \times 10^{23}\, mol^{-1}$$

4

4

The phase space

- If N is the number of particles of a given classical system, the specification of a microstate requires the specification of 3N position coordinates $q_1, q_2, \ldots, q_{3N}$ and 3N momentum coordinates $p_1, p_2, \ldots, p_{3N}$
- Geometrically, the set of coordinates $(q,p) \equiv \{(q_i,p_i)\}_{i=1, 2, \ldots, 3N}$, may be regarded as a point in a space of 6N dimensions. We refer to this space as the phase space Γ, and the phase point (q,p) as a representative point, of the given system in Γ
- Of course, the coordinates q_i and p_i are functions of the time t; the precise manner in which they vary with t is determined by the canonical equations of motion,

  where $H(q_i,p_i)$ is the Hamiltonian of the system

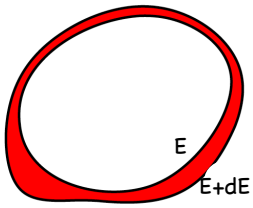
$$\left.\begin{aligned} \dot{q}_i &= \frac{\partial H(q_i,p_i)}{\partial p_i} \\ \dot{p}_i &= -\frac{\partial H(q_i,p_i)}{\partial q_i} \end{aligned}\right\} \quad i = 1,2,\cdots,3N$$

- Now, as time passes, the set of coordinates (q,p) undergoes a continual change. Correspondingly, our representative point draw a trajectory in Γ

5

Probabilistic description

- Note that the initial microstate of a macroscopic system in unknowable!
- We can collect information only on a small number of quantities, like energy ⇒ probabilistic description
- If, for example, the total energy of the system is known to have a fixed value, say E, the corresponding trajectory will be restricted to the "hypersurface" $H(q_i,p_i)=E$ of the phase space
- On the other hand, if the total energy may lie anywhere in the range (E, E+dE), the corresponding trajectory will be restricted to the "hyper-shell" defined by these limits ... and so on for other experimental conditions



- In this and other experimental conditions, one could postulate that for a given macrostate, a statistical system, at any time t, is equally likely to be in any one of an extremely large number of distinct microstates

6

Ensemble theory

- It may, therefore, make sense if we consider, at a single instant of time, a rather large number of systems (all being some sort of different preparations of the given system) which are characterized by the same macrostate as the original system but are in all sorts of possible microstates.
- Then, under ordinary circumstances, we may expect that the average behaviour of any system in this collection, which we call an ensemble, would be identical with the time-averaged behaviour of the given system.
- It is on the basis of this expectation that we proceed to develop the so-called ensemble theory
- In the phase space, the corresponding picture will consist of a collection of representative points, one for each member of the ensemble, all lying within the "allowed" region of this space … a probability density!

7

7

Ensemble theory

8

- Statistical mechanics is thus based on the Gibbs ensemble concept. That is, many microscopic configurations lead to the same macroscopic properties, implying that it is not necessary to know the precise detailed motion of every particle in a system in order to predict its properties. It is sufficient to simply average over a large number of identical systems, each in a different microscopic configuration; i.e., the macroscopic observables of a system are formulated in terms of ensemble averages.

$$\langle f \rangle_{p(q,p)} = \int d^{3N}q d^{3N}p \; f(q,p)\; p(q,p)$$

- However, this is not the way we usually think about the average behavior of a system. In most experiments we perform a series of measurements during a certain time interval and then determine the average of these measurements
- To take a specific example, let us consider a fluid consisting of atoms. Suppose that we wish to compute the average value of a property f(q,p) that, in principle, could depend on the coordinates and momenta of all particles in the system
- As time progresses, the atomic coordinates will change (according to Newton's equations of motion), and hence this property will change.

8

Time averages vs initial conditions

- Provided that at time t_0 we have prepared the macroscopic/complex system in a specific (unknown) microscopic state (initial coordinates and momenta of all atoms: $[q(t_0),p(t_0)]$), in an experiment we measure a time-average:

$$\langle f \rangle_{\text{exp.}} \underset{\Delta t >> 1}{=} \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} dt\, f(q(t), p(t))$$

- Note that, in writing down this equation, we have implicitly assumed that, for Δt sufficiently long, the time average does not depend on the initial conditions. This is, in fact, a subtle assumption that is not true in general
- However, we shall disregard subtleties and simply we have to assume that time averages do not depend on the initial coordinates and momenta. If that is true, then we would not change our result if we average over many different initial conditions

$$\left\langle \langle f \rangle_{\text{exp.}} \right\rangle_{init.} \underset{N_{init.}>>1}{=} \frac{1}{N_{init.}} \sum_{j=1}^{N_{init.}} \frac{1}{\Delta t} \int_{t_0}^{t_0+\Delta t} dt\, f(q(t,j), p(t,j)) \underset{!}{\equiv} \langle f \rangle_{\text{exp.}}$$

- This is true if averaging over all initial phase space coordinates (q,p) , which are compatible with the experimental constraints, is equivalent to averaging over the time evolved phase space coordinates.

9

Ergodic hypothesis

- The ensemble averages at equilibrium do not depend on time
- This is so, because there is (it can be assumed) a one-to-one correspondence between the initial phase space coordinates of a system and those that specify the state of the system at a later time

$$\langle f \rangle_{p(q,p)} = \int d^{3N}q\, d^{3N}p\, f(q,p)\, p(q,p) \Leftrightarrow \left\langle \langle f \rangle_{\text{exp.}} \right\rangle_{init.} \underset{!}{\equiv} \langle f \rangle_{\text{exp.}}$$

- in what follows, we shall simply assume that the "ergodic hypothesis", as is usually referred to, applies to the systems that we will study

$$\langle f \rangle_{p(q,p)} = \lim_{N_{init.}\to\infty} \left\langle \langle f \rangle_{\text{exp.}} \right\rangle_{init.} = \lim_{\Delta t\to\infty} \langle f \rangle_{\text{exp.}}$$

- The student, however, should be aware that many examples of systems are not ergodic in practice … such as, for example, glasses and metastable phases

10

The density/distribution function

- As time passes, every member of the ensemble undergoes a continual change of microstates; correspondingly, the representative points constituting the collection continually move along their respective trajectories
- The overall picture of this movement possesses some important features which are best illustrated in terms of what we call a density function $\rho(q,p,t)$ or distribution function of the phase space.
- This function is such that, at any time t, the number of representative points in the "volume element" $(d^{3N}q d^{3N}p)$ around the point (q,p) of the phase space is given by the product $\rho(q,p,t)d^{3N}q\, d^{3N}p$, which therefore corresponds to a density function proportional to the probability that the microstate of the system is in $(d^{3N}q d^{3N}p)$ around the point (q,p) at time t
- Clearly, the density function $\rho(q,p,t)$ symbolizes the manner in which the members of the ensemble are distributed over all possible microstates at different instants of time

11

The ensamble average

12

- Accordingly, given a particular density function $\rho(q,p,t)$, the ensemble average of a given physical quantity f(q,p) would be given by

$$\left\langle f(q,p)\right\rangle_{\rho(q,p,t)} = \frac{\int f(q,p)\,\rho(q,p,t)\,d^{3N}q\,d^{3N}p}{\int \rho(q,p,t)\,d^{3N}q\,d^{3N}p}$$

[here I have assumed that $\rho(q,p,t)$ have to be normalized to become a probability p(q,p,t)] the integrations extend over the whole of the phase space; however, only the populated regions of the phase space ($\rho(q,p,t)\neq 0$) really contribute.

- In general, the ensemble average $\langle f\rangle$ may itself be a function of time. An ensemble is said to be stationary if $\rho(q,p,t)$ does not depend explicitly on time, i.e. at all times $\rho(q,p,t) = \rho(q,p)$
- Clearly, for such an ensemble the average value of any physical quantity f(q,p) will be independent of time.
- A stationary ensemble qualifies to represent a system in equilibrium. To determine the circumstances under which this may hold, we have to study the movement of the representative points in the phase space.

Liouville's theorem

- Since the total number of systems in an ensemble is conserved, the number of representative points leaving any volume in Γ space per second must be equal to the rate of decrease of the number of representative points in the same volume.
- By using the divergence theorem, over the flux of representative points in/out of a particular volume in Γ space across its bounding surface, one can easily obtain a equation of continuity for the swarm of the representative points (see supplementary material for more details):

$$\frac{\partial \rho(q,p,t)}{\partial t} + \vec{\nabla} \cdot \left[\rho(q,p,t) \vec{v} \right] = 0$$

- Which, by making explicit the divergence, can be easily rewritten in the following form: Poisson's brackets

$$\frac{d\rho}{dt} = \frac{\partial \rho}{\partial t} + \{\rho, H\} = \frac{\partial \rho}{\partial t} + \sum_i \left(\frac{\partial \rho}{\partial q_i}\frac{\partial H}{\partial p_i} - \frac{\partial \rho}{\partial p_i}\frac{\partial H}{\partial q_i} \right) = 0$$

- This is the so-called Liouville 's theorem (1838), which says that the "local" density of the representative points stays constant in time

13

Equal a priori probabilities

- Thus, the swarm of the representative points moves in the phase space in essentially the same manner as an incompressible fluid moves in the physical space!
- The condition which ensures that ρ(q,p,t) is stationary is clearly:

$$\{\rho, H\} = 0 \quad \Rightarrow \quad \frac{\partial \rho}{\partial t} = 0$$

- Now, one possible way of satisfying this is to assume that ρ, which is already assumed to have no explicit dependence on time, is independent of the coordinates (q,p) as well, i.e. ρ(q,p)=const. over the relevant region Ω of the phase space (and, of course, zero everywhere else).
- Physically, this choice corresponds to an ensemble of systems which at all times are uniformly distributed over all possible microstates. The ensemble average then reduces to

$$\langle f \rangle = \frac{\int_\Omega f(q,p)(const.) d^{3N}q d^{3N}p}{\int_\Omega (const.) d^{3N}q d^{3N}p} = \frac{1}{\int_\Omega d^{3N}q d^{3N}p} \int_\Omega f(q,p) d^{3N}q d^{3N}p$$

where $\int_\Omega d^{3N}q d^{3N}p$ is the total volume of the relevant region Ω

14

Ensamble Microcanonico: probabilità di ogni composizione è la stessa

funzione di partizione: dimensione di volume nello spazio delle fasi

The microcanonical ensemble

15

- Clearly, in this case, any member of the ensemble is equally likely to be in any one of the various possible microstates. This statement is usually referred to as the postulate of "equal a priori probabilities" for the various possible microstates; the resulting ensemble is referred to as the microcanonical ensemble.
- To be specific, consider a simple fluid with N particles in a box V

  Postulate I: the relevant constants of motion are the energy E, the total linear momentum **P** and the total angular momentum **L** , but if the system is in a box fixed in our laboratory then **P**=**L**=0, thus

  $$\rho(q,p) = f\big(H(q,p)\big)$$

  Postulate II: the microstates with equal energy E have the same probability (microcanonical ensemble)

  $$\rho(q,p) = \begin{cases} const. & if\ E < H(q,p) < E + \Delta \quad (\Delta << E) \\ 0 & otherwise \end{cases}$$

- Let us assume that the normalization of the density function is

  $$\int \frac{d^{3N}q\, d^{3N}p}{h^{3N}N!}\rho(q,p) = 1 \quad \Rightarrow \quad const. = \frac{1}{\Omega(E,N,V,\Delta)} \quad with \quad \Omega(E,N,V,\Delta) = \int_{E<H<E+\Delta} \frac{d^{3N}q\, d^{3N}p}{h^{3N}N!}$$

  h=6.626×10^{-34} Js is the Plank's constant.

15

N!
indistinguibilità delle particelle
non da un punto di vista quantistico
ma da uno dal nostro di simulazione

Computer simulations

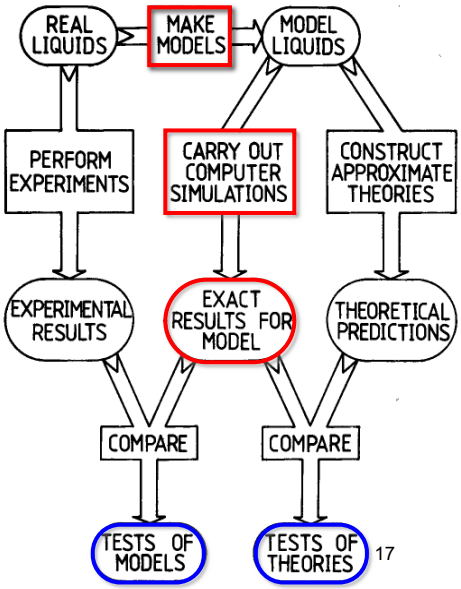
- Some problems in statistical mechanics are exactly soluble. By this, we mean that a complete specification of the microscopic properties of a system (such as the Hamiltonian of an idealized model like the perfect gas or the Einstein crystal) leads directly, and perhaps easily, to a set of interesting results or macroscopic properties (such as an equation of state like pV = Nk$_B$T)
- There are only a handful of non-trivial, exactly soluble problems in statistical mechanics; the two-dimensional Ising model is a famous example.
- Some problems in statistical mechanics, while not being exactly soluble, succumb readily to analysis based on a straightforward approximation scheme. Computers may have an incidental, calculational, part to play in such work.
- The problem is that many "straightforward" approximation schemes simply do not work when applied to complex systems. For some systems, it may not even be clear how to begin constructing an approximate theory in a reasonable way

16

16

- The more difficult and interesting the problem, the more desirable it becomes to have exact results available, both to test existing approximation methods and to point the way towards new approaches.

- Computer simulations have a valuable role to play in providing essentially exact results for problems in statistical mechanics
- Computer simulation is a test of theories and, historically, simulations have indeed discriminated between well-founded approaches and ideas that are plausible but, in the event, less successful
- The results of computer simulations may also be compared with those of real experiments. In the first place, this is a test of the underlying model used in a computer simulation



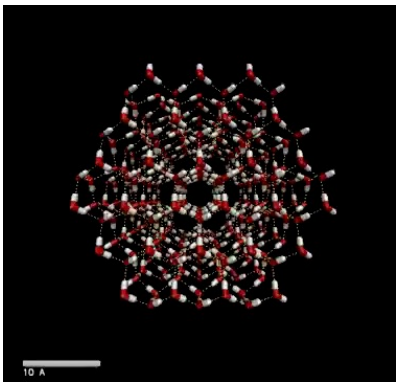
17

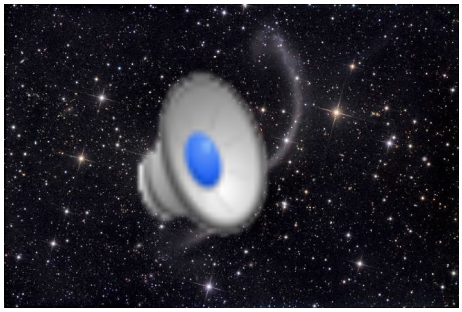
17

Computer experiments

18

- Eventually, if the model is a good one, the simulator hopes to offer insights to the experimentalist, and assist in the interpretation of new results. These techniques are often termed "computer experiments"
- A wide range of physical phenomena, from the molecular scale to the galactic, may be studied using some form of computer simulation





18

- Computer simulation provides a direct route from the microscopic details of a system (the masses of the atoms, the interactions between them, molecular geometry etc.) to macroscopic properties of experimental interest
- As well as being of academic interest, this type of information is technologically useful. It may be difficult or impossible to carry out experiments under extreme conditions
- But note that computers offers us no understanding, only numbers. And, as in a real experiment, these numbers have statistical errors.
- So what we get out of a simulation is never directly a theoretical relation. As in a real experiment, we still have to extract the useful information.
- Thus it is dangerous to say that through simulations we can prove something … even if sometimes we are tempted to say this.

19

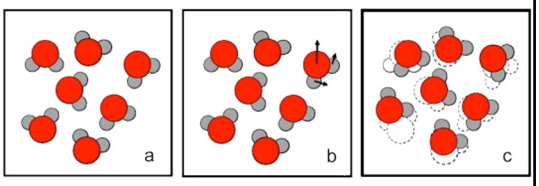
19

Introduction to Molecular Dynamics

- Molecular dynamics (MD) is the science of simulating the time dependent behavior of a system of particles. The time evolution of the set of interacting atoms is followed by integrating their equation of motion with boundary conditions appropriate for the geometry or symmetry of the system

$$\left.\begin{aligned}\dot{q}_i &= \frac{\partial H(q_i,p_i)}{\partial p_i}\\ \dot{p}_i &= -\frac{\partial H(q_i,p_i)}{\partial q_i}\end{aligned}\right\} \quad i = 1,2,\cdots,3N$$

- Molecular dynamics generate information at the microscopic level, which are: atomic positions and velocities



- In order to calculate the microscopic behavior of a system from the laws of classical mechanics, MD requires, as an input, a description of the interaction potential. The quality of the results of an MD simulation depends on the accuracy of the description of inter-particle interaction potential

20

20

- Thus MD technique acts as a *computational microscope*. This microscopic information is then converted to the macroscopic observable like pressure, temperature, heat capacity and stress tensor etc. using statistical mechanics

- When we perform a real experiment, we prepare a sample of the material, we connect this sample to a measuring instrument, and we measure the property of interest during a certain time interval

- Our measurements are subject to statistical noise, the longer we average, the more accurate our measurement becomes.

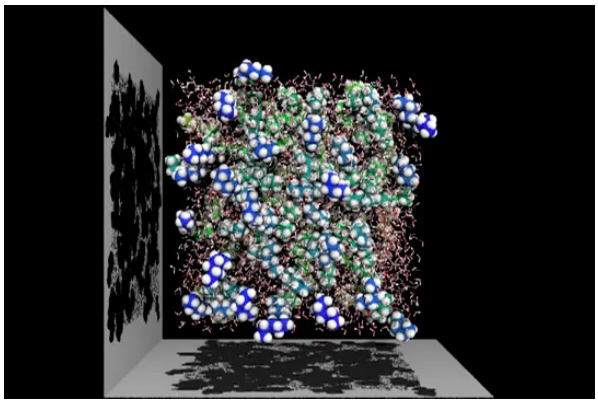
- Molecular Dynamics simulations are in many respects very similar to real experiments:

1. First, we prepare a sample: we select a model system consisting of N particles and we solve Newton's equations of motion for this system until the properties of the system no longer change with time (we equilibrate the system)
2. After equilibration, we perform the actual measurement.

21

21

- In fact, some of the most common mistakes that can be made when performing a computer experiment are very similar to the mistakes that can be made in real experiments (e.g., the sample is not prepared correctly, the measurement is too short, the system undergoes an irreversible change during the experiment, etc.)



- These situations, however, not necessarily represent a "mistake": Molecular Dynamics simulations can be used, in fact, to study out of equilibrium systems not only equilibrium statistical properties

22

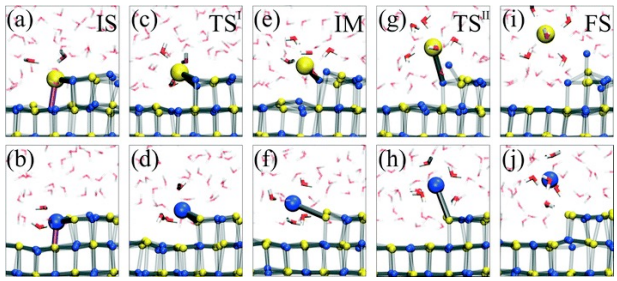
22

Ab-initio Molecular Dynamics

23

- The disadvantage of a model interaction is that a system is restricted to a single molecular connectivity. This prohibits interaction models from describing chemical processes involving bond breaking and forming.
- An alternative approach is the combination of classical dynamics with electronic structure: inter-nuclear forces are computed *on the fly* from an electronic structure calculation as a MD simulation proceeds. This method, known as ab-initio molecular dynamics, requires no input potential model and is capable of describing chemical events, although it has high computational overhead … and it's unavoidably build upon some approximations
- We will focus on "exact" simulation approaches in statistical mechanics

[NaCl dissolution] top row: initial dissolution of Cl ion; bottom row: Na ion. (a) and (b): the initial state (IS); (c) and (d): first transition state (TS^{I}); (e) and (f): intermediate states (IM); (g) and (h): second transition state (TS^{II}); (i) and (j): final states (FS), fully solvated ions



23

Chaos!

- But any two classical trajectories which are initially very close will eventually diverge from one another exponentially with time (Lyapunov instability)
- In the same way, any small perturbation, even the tiny error associated with finite precision arithmetic, will tend to cause a computer-generated trajectory to diverge from the true classical trajectory with which it is initially coincident.
- A small perturbation applied at time t = 0 causes the trajectories in the perturbed simulation to diverge from the reference trajectories, and become statistically uncorrelated, within a few hundred time steps.
- One can consider the growing average "distance in configuration space" $|\Delta \mathbf{r}|$, defined via

$$\left|\Delta\vec{r}\right|^2 = \frac{1}{N}\sum_{i=1}^{N}\left|\vec{r}_i(t) - \vec{r}_i^{\,0}(t)\right|^2$$

$\mathbf{r}_i^0(t)$ being the position of molecule i at time t in a reference simulation, and $\mathbf{r}_i(t)$ being the position of the same molecule at the same time in the perturbed simulation.

24

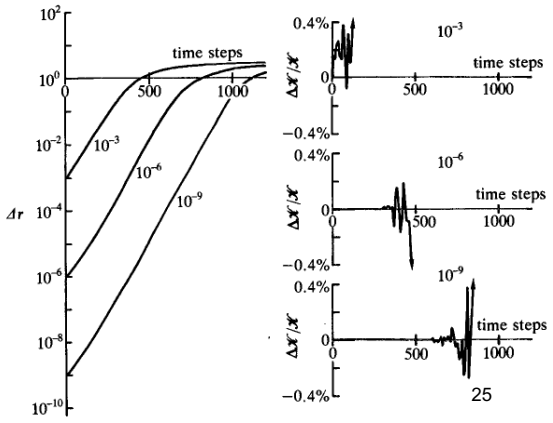
24

non è così importante
avere una traiettoria esatta
(nel diagramma delle fasi)
piuttosto avere fissata
Energia pressione temp
o quelle che sono le variabili
fissate

in particolare
vogliamo algoritmi che siano
bravi a tenere l'energia

Per tempi brevi si deve
comunque garantire
la traiettoria

- In the three cases illustrated here, all the molecules in the perturbed runs are initially displaced in random directions from their reference positions at t = 0, by 10^{-3} σ, 10^{-6} σ, and 10^{-9} σ, respectively, where σ is the molecular diameter. In all other respects, the runs are identical
- As the runs proceed, however, other mechanical quantities eventually become statistically uncorrelated. In figure, we show also the percentage difference in kinetic energies between perturbed and reference simulations.
- On the scale of the figure, the kinetic energies remain very close for a period whose length depends on the size of the initial perturbation; after this point the differences become noticeable very rapidly.



25

- Presumably, both the reference trajectory and the perturbed trajectory are diverging from the true solution of Newton's equations.
- This so-called Lyapunov instability would seem to be a devastating blow to the whole idea of Molecular Dynamics simulations but we have good reasons to assume that even this problem need not be serious.
- Clearly, no integration algorithm will provide an essentially exact solution for a very long time. Fortunately, we do not need to do this. Remember that molecular dynamics serves two roles:

1. Firstly, we need essentially exact solutions of the equations of motion for times comparable with the correlation times of interest, so that we may accurately calculate time correlation functions (if we need them).
2. Secondly, we use the method to generate states sampled from the micro-canonical ensemble. We do not need exact classical trajectories to do this, but must lay great emphasis on energy conservation as being of primary importance for this reason. The point is that the particle trajectories must stay on the appropriate constant-energy hyper-surface in phase space, otherwise correct ensemble averages will not be generated.

26

26

Integration algorithms

- Energy conservation is an important criterion, but actually we should distinguish two kinds of energy conservation, namely, short time and long time.
- The sophisticated higher-order algorithms tend to have very good energy conservation for short times. However, they often have the undesirable feature that the overall energy drifts for long times.
- Energy conservation is degraded as the time step is increased, and so all simulations involve a trade-off between economy and accuracy: a good algorithm permits a large time step to be used while preserving acceptable energy conservation.
- Other factors dictating the energy-conserving properties are the shape of the potential energy curves and the typical particle velocities. Thus, shorter time steps are used at high temperatures, for light molecules, and for rapidly varying potential functions

27

27

Numerical Integration Techniques

- The potential energy is a function of the atomic positions (3N) of all the atoms in the system. Due to the complicated nature of this function, there is no analytical solution to the equations of motion and these equation must be solved numerically.
- Numerous numerical algorithms have been developed for integrating the equations of motion. We list several here.
  - (i) Verlet algorithm, (for the others see supplementary mat.)
  - (ii) Leap-frog algorithm,
  - (iii) Velocity Verlet,
  - (iv) Beeman's algorithm and
  - (v) Symplectic reversible integrators,
- In choosing which algorithm to use, one considers the following criteria:
  - (i) The algorithm should conserve energy and momentum and is reversible. When $\delta t \rightarrow -\delta t$ the system should go back to original state.
  - (ii) It should be computationally efficient.
  - (iii) It should permit a long time step for integration.
  - (iv) Only one force evaluation per time step (important for complex potential).

28

->inversione temporale è l'egata con la conserva zione dell'energia

28

Verlet algorithm

29

- The most widely used finite-difference method is a third-order "Taylor expansion" algorithm first used by Verlet (1967) and widely known as the Verlet's method. It is derived from the two Taylor expansion

$$\vec{r}(t+\delta t) = \vec{r}(t) + \delta t\vec{v}(t) + \frac{1}{2}\delta t^2\vec{a}(t) + \frac{1}{3!}\delta t^3\dot{\vec{a}}(t) + O(\delta t^4)$$

$$\vec{r}(t-\delta t) = \vec{r}(t) - \delta t\vec{v}(t) + \frac{1}{2}\delta t^2\vec{a}(t) - \frac{1}{3!}\delta t^3\dot{\vec{a}}(t) + O(\delta t^4)$$

summing the above two equations eliminates the odd-order terms.

$$\vec{r}(t+\delta t) + \vec{r}(t-\delta t) = 2\vec{r}(t) + \delta t^2\vec{a}(t) + O(\delta t^4)$$

$$\Rightarrow \boxed{\vec{r}(t+\delta t) \cong 2\vec{r}(t) - \vec{r}(t-\delta t) + \delta t^2\vec{a}(t)}$$

noto: non c'è la velocità è reversibile nel tempo (delta t)

- Notice that the position vector r at time t+δt is calculated from position vector at time t and t-δt, this makes the Verlet's algorithm a two-step method. Therefore it is not self-starting, initial positions r(0) and velocities v(0) are not sufficient to begin a calculation.
- Another observation regarding the Verlet algorithm is that it is properly centred, i.e. r(t-δt) and r(t+δt) play symmetrical roles making it time-reversible.

29

- Verlet algorithm does not use the velocity to compute the new position. One, however, can derive the velocity from knowledge of the trajectory, using

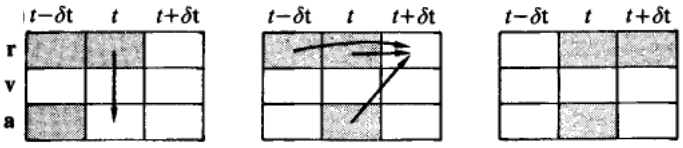
$$\vec{r}(t+\delta t) - \vec{r}(t-\delta t) = 2\delta t\vec{v}(t) + O(\delta t^3)$$

or

$$\vec{v}(t) = \frac{\vec{r}(t+\delta t) - \vec{r}(t-\delta t)}{2\delta t} + O(\delta t^2) \cong \frac{\vec{r}(t+\delta t) - \vec{r}(t-\delta t)}{2\delta t}$$

se volessimo la velocità avremmo meno precisione

- This expression for the velocity is only accurate to order δt^2 . So, in its original form, it treats velocity as less important than positions. The overall scheme of this algorithm is illustrated in this figure:

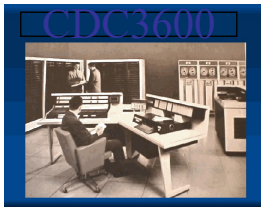


- The Verlet method requires essentially 9N words of storage, making it very compact, and it is simple to program. It has been shown to have excellent energy-conserving properties even with long time steps.

30

30

Molecular Dynamics history



- 1957 (Alder & Wainwright) hard-spheres liquid-solid phase transition (UNIVAC)
- 1960 (Gibson et al.) study of the creations of defects under radiation damage with continuous potential (IBM 704)
- 1964 (Rahman) liquid Argon studied with a Lennard-Jones potential (CDC 3600)
- 1967 (Verlet) Argon phase diagram with Lennard-Jones potential
- Nowadays molecular dynamic techniques have been widely used in many branches of science: statistical mechanics, chemistry to biophysics, solid state physics and material science, and so on.

31

31

Studying small systems

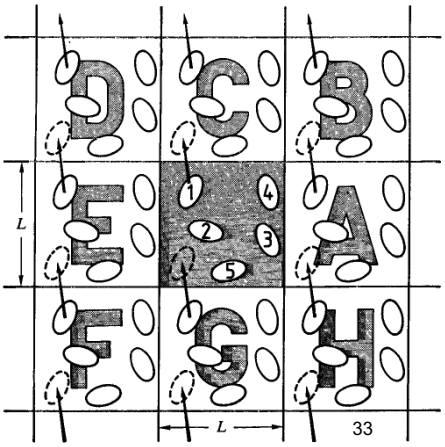
- Computer simulations are usually performed on a small number of molecules, $N < 10^5$: the size of the system is limited by the available storage on the host computer, and, more crucially, by the speed of execution of the program
- The time taken for a double loop used to evaluate the (pairwise) forces or potential energy is proportional to N^2. Special techniques may reduce this dependence to $O(N)$, for very large systems; clearly, smaller systems will always be less expensive
- If we are interested in the properties of a very small liquid drop, or a microcrystal, then the simulation will be straightforward. The cohesive forces between molecules may be sufficient to hold the system together during the course of a simulation
- Otherwise our set of N molecules may be confined by a potential representing a container, which prevents them from drifting apart. These arrangements, however, are not satisfactory for the simulation of bulk systems.

32

32

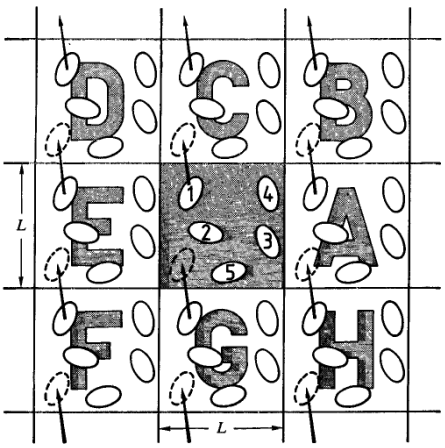
Periodic Boundary Conditions (pbc)

- A major obstacle to such a simulation is the large fraction of molecules which lie on the surface of any small sample; for 10^3 molecules arranged in a 10x10x10 cube, no less than 488 molecules appear on the cube faces. Whether or not the cube is surrounded by a containing wall, molecules on the surface will experience quite different forces from molecules in the bulk.
- The problem of surface effects can be overcome by implementing periodic boundary conditions: the cubic box is replicated throughout space to form an infinite lattice
- In the course of the simulation, as a molecule moves in the original box, its periodic image in each of the neighbouring boxes moves in exactly the same way. Thus, as a molecule leaves the central box, one of its images will enter through the opposite face



33

- There are no walls at the boundary of the central box, and no surface molecules. This box simply forms a convenient axis system for measuring the coordinates of the N molecules
- A two-dimensional version of such a periodic system is shown in the picture. The duplicate boxes are labelled A, B, C, etc. in an arbitrary fashion.
- As particle 1 moves through a boundary, its images, 1A, 1B, etc. (where the subscript specifies in which box the image lies) move across their corresponding boundaries.



- The number density in the central box (and hence in the entire system) is conserved. It is not necessary to store the coordinates of all the images in a simulation (an infinite number!), just the molecules in the central box. When a molecule leaves the box by crossing a boundary, attention may be switched to the image just entering

34

34

problema:
potenziali per r grandi
densità non conservate

- It is important to ask if the properties of a small, infinitely periodic, system, and the macroscopic system which it represents, are the same. This will depend both on the range of the intermolecular potential and the phenomenon under investigation
- For a fluid of Lennard-Jones atoms (atom-atom interaction potential: $v(r)=4\varepsilon[(\sigma/r)^{12}-(\sigma/r)^{6}]$) or any other short range potential, it should be possible to perform a simulation in a box with p.b.c, without a particle being able to "sense" the symmetry of the periodic lattice.
- If the potential is long ranged (i.e. $v(r) \approx r^{-\nu}$ where ν is less than the dimensionality of the system) there will be a substantial interaction between a particle and its own images in neighbouring boxes, and consequently the symmetry of the cell structure is imposed on a fluid which is in reality isotropic. There are methods used to cope with long-range potentials (e.g. the "Ewald sum" for the Coulomb potential)
- The use of periodic boundary conditions inhibits the occurrence of long-wavelength fluctuations. For a cube of side L, the periodicity will suppress any density waves with a wavelength greater than L. Thus, it would be delicate to simulate a liquid close to the gas-liquid critical point, where the range of critical fluctuations is macroscopic.

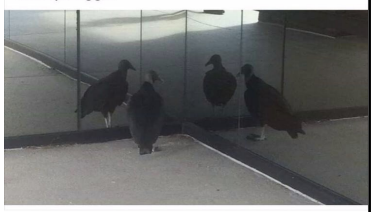
35

35

36

- The same limitations apply to the simulation of long-wavelength phonons in model solids, where, in addition, the cell periodicity picks out a discrete set of available wave-vectors (i.e. $\mathbf{k}= (k_x, k_y, k_z)2\pi/L$, where k_x, k_y and k_z are integers) in the first Brillouin zone.
- Periodic boundary conditions have also been shown to affect the rate at which a simulated liquid nucleates and forms a solid or glass when it is rapidly cooled.
- Despite the above remarks, the common experience in simulation work is that periodic boundary conditions have little effect on the equilibrium thermodynamic properties and structures of fluids away from phase transitions and where the interactions are short-ranged. It is always sensible to check that this is true for each model studied
- If the resources are available, it should be standard practice to increase the number of molecules (and the box size, so as to maintain constant density) and rerun the simulations

"Vi starete chiedendo perché vi ho convocati tutti qui oggi".



36

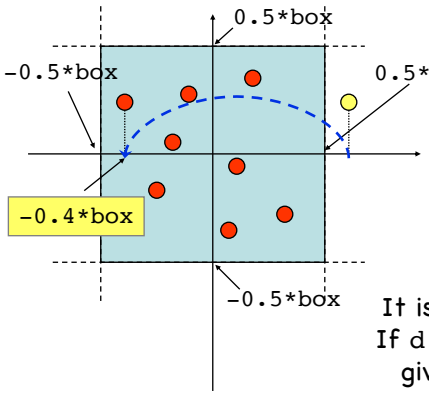
bias 1: PBC

Periodic Boundary Conditions algorithm
(and minimum image convention)

37

$$Pbc(r) = r - box*rint(r/box)$$

Rint(arg) = nearest (signed) integer to arg;
Ex.: rint(-0.55) = -1
rint(0.1) = 0



r=0.6*box
⇒ Pbc(r)=0.6*box-box*rint(0.6)=
=0.6*box-box=-0.4*box

It is used also for distances (minimum image):
If d >0.5*box or <-0.5*box the algorithm gives the distance with the nearest image!

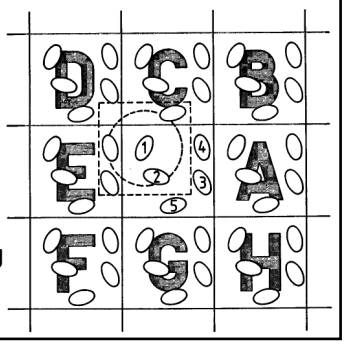
37 convenzione della minima immagine:
se calcolo la distanza mi valuta la distanza dell'immagine nella cellula + vicina.

Bias 2:

Potential energy

38

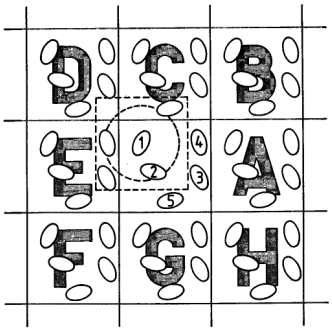
- Now we must turn to the question of calculating properties of systems subject to periodic boundary conditions. The heart of simulation programs involves the calculation of the potential energy of a particular configuration.
- With the Metropolis Monte Carlo (MC) algorithm (we shall see that) the potential energy V is part of the configurational probability density we wish to sample (e.g. p=exp(-βV)/Z for the canonical ensemble)
- With the Molecular Dynamics (MD) method, where one solves Newton's equations of motion for an assembly of N particles, the potential energy is needed to compute the forces acting on such particles (F=-∇V)
- Consider how we would calculate the force on molecule 1, or those contributions to the potential energy involving molecule 1, assuming pairwise additivity



38

Minimum image convention

- We must include interactions between molecule 1 and every other molecule i in the simulation box



- There are N-1 terms in this sum. However, in principle, we must also include all interactions between molecule 1 and images iA, iB, etc. lying in the surrounding boxes. This is an infinite number of terms, and of course is impossible to calculate in practice
- For a short-range potential energy function, we may restrict this summation by making an approximation: Consider molecule 1 to rest at the centre of a region which has the same size and shape as the basic simulation box (see the figure)
- Molecule 1 interacts with all the molecules whose centres lie within this region, that is with the closest periodic images of the other N-1 molecules. This is called the "minimum image convention": for example, in the figure molecule 1 interacts with molecules 2,3E, 4E and 5C.

39

39

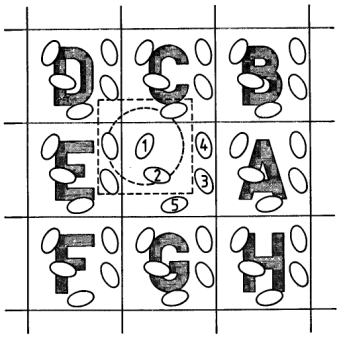
Bias 3:

interazione anisotropa
a seconda se è lungo
il lato o la bisettrice

Nuovo bias: Cutoff

Potential cut-off

- In the minimum image convention, then, the calculation of the potential energy due to pairwise-additive interactions involves N(N-1)/2 terms.
- A further approximation significantly improves this situation. The largest contribution to the potential and forces comes from neighbours close to the molecule of interest, and for short-range forces we normally apply a spherical cutoff.

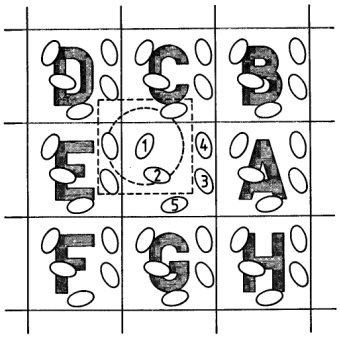


- This means setting the pair potential v(r) to zero for $r > r_c$, where r_c is the cutoff distance. The dashed circle in figure represents a cutoff, and in this case molecules 2 and 4E contribute to the force on 1, since their centres lie inside the cutoff, whereas molecules 3E and 5C do not contribute
- The introduction of a spherical cutoff should be a small perturbation, and the cutoff distance should be sufficiently large to ensure this.

40

40

- Of course, the penalty of applying a spherical cutoff is that the thermodynamic (and other) properties of the model fluid will no longer be exactly the same as for (say) the non-truncated, Lennard-Jones fluid



- As we shall see, it is possible to apply long-range corrections to such results, so as to recover, approximately, the desired information.
- The cutoff distance must be no greater than L/2 for consistency with the minimum image convention
- The simplest method to truncate potentials is to ignore all interaction beyond r_c. This may result in an appreciable error in our estimate of the potential energy of the true potential.
- Usually, it is simply assumed that we can correct for the truncation of the intermolecular potential by adding a "tail corrections" to the potential energy and to other quantities influenced by the cut-off, e.g. pressure

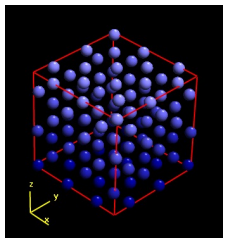
41

41

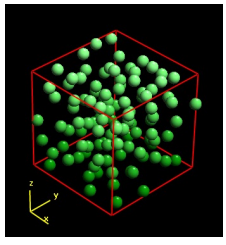
42

Initialization

- To start the simulation, we should assign initial positions to all particles in the system. As the equilibrium properties of the system do not (or, at least, should not) depend on the choice of initial conditions, all reasonable initial conditions are in principle acceptable
- If we wish to simulate the solid state of a particular model system, it is logical to prepare the system in the crystal structure of interest.



- In contrast, if we are interested in the fluid phase, we simply prepare the system in any convenient crystal structure (random configurations must be avoided because easily give superposition of particles!)
- This crystal subsequently melts during the initial stages of the simulation, because the solid state is not thermodynamically stable at the given T



42

non fare partire con particelle sovrapposte: pensare potenziale L-J

PARTIRE DA RETICOLO CRISTALLINO

unità di misure privilegiate per le simulazioni

Reduced Units

- In simulations it is often convenient to express quantities such as temperature, density, pressure etc. in reduced units. This means that we choose a convenient unit of energy, length and mass and then express all other quantities in terms of these basic units.
- In the example of a Lennard-Jones system, a natural choice is following:
  (1) Unit of length, σ (2) Unit of energy, ε (3) Unit of mass, m

  and from these basic units, all other units follow. For instance, our unit of time is $\sigma\sqrt{(m/\varepsilon)}$ and the unit of temperature is ε/k_B
- In terms of these reduced units, denoted with superscript *, the reduced pair potential is a dimensionless function of the reduced distance $r^*=r/\sigma$. For instance, the reduced form for the Lennard-Jones potential is $v^*(r)=4[(1/r^*)^{12}-(1/r^*)^6]$
- The most important reason to introduce reduced units is that (infinitely) many combinations of ρ, T, ε and σ all correspond to the same state in reduced units. This is the law of corresponding states

43

43

- the same simulation of a LJ model can be used to study Ar at 60 K and a density of 840 kg/m^3 and Xe at 112 K and a density of 1617 kg/m^3 . In reduced units, both simulations correspond to the state point ρ^*= 0.5, T*=0.5. If we had not used reduced units, we might have easily missed the equivalence of these two simulations.
- With these conventions we can define the following reduced units: the potential energy $V^*= V/\varepsilon$, the pressure $P^*=P\sigma^3/\varepsilon$, the density $\rho^*=\rho\sigma^3$, and the temperature $T^* = k_BT/\varepsilon$.
- Another, practical, reason for using reduced units is the following: when we work with real (SI) units, we find that the absolute numerical values of the quantities that we are computing are either much less or much larger than 1. If we multiply several such quantities using standard floating-point multiplication, we face a distinct risk that, at some stage, we will obtain a result that creates an overflow or underflow
- Conversely, in reduced units, almost all quantities of interest are of order 1 (say, between 10^{-3} and 10^3). Hence, if we suddenly find a very large (or very small) number in our simulations, then there is a good chance that we have made an error somewhere. In other words, reduced units make it easier to spot errors

44

44

Temperature and Pressure

- The temperature (for MD simulations) and pressure may be calculated using the virial theorem, which we can write in the form of "generalized equipartition" [Münster 1969]: for any generalized coordinate q_k or momentum p_k.

$$\left\langle p_k \frac{\partial H}{\partial p_k} \right\rangle = k_B T \quad ; \quad \left\langle q_k \frac{\partial H}{\partial q_k} \right\rangle = k_B T$$

- These equation may be easily derived in the canonical ensamble. The first equation is particularly simple when the momenta appear as squared terms in the Hamiltonian. For example, in the atomic case, we may sum up 3N terms of the form p^2/m to obtain
-

$$\left\langle \sum_{i=1}^{N} \left|\vec{p}_i\right|^2 / m_i \right\rangle = 2\langle K \rangle = 3Nk_B T$$

  This is the familiar equipartition principle: an average energy of $k_B T/2$ per degree of freedom.

- The pressure may be calculated using the second equation.

45

45

- If we choose Cartesian coordinates, and use Hamilton's equations of motion, it is easy to see that each coordinate derivative in this equation is the negative of a component of the force $\mathbf{f}_i$ on some particle i, and we may write, summing over N particles,

$$-\frac{1}{3}\left\langle \sum_{i=1}^{N} \vec{r}_i \cdot \vec{\nabla}_i V \right\rangle = \frac{1}{3}\left\langle \sum_{i=1}^{N} \vec{r}_i \cdot \vec{f}_i^{tot} \right\rangle = -Nk_B T$$

- We have used the symbol $\mathbf{f}^{tot}$ because this represents the sum of intermolecular forces and external forces. The latter are related to the external pressure, as can be seen by considering the effect of the container walls on the system:

$$\frac{1}{3}\left\langle \sum_{i=1}^{N} \vec{r}_i \cdot \vec{f}_i^{ext} \right\rangle = -PV$$

- By restricting the attention to inter-particle forces, we define the "virial" W as

$$-\frac{1}{3}\sum_{i=1}^{N} \vec{r}_i \cdot \vec{\nabla}_i V = \frac{1}{3}\sum_{i=1}^{N} \vec{r}_i \cdot \vec{f}_i = W$$

  then

$$0 = \frac{1}{3}\left\langle \sum_{i=1}^{N} \vec{r}_i \cdot \vec{f}_i^{tot} \right\rangle + Nk_B T = \frac{1}{3}\left\langle \sum_{i=1}^{N} \vec{r}_i \cdot \vec{f}_i \right\rangle + \frac{1}{3}\left\langle \sum_{i=1}^{N} \vec{r}_i \cdot \vec{f}_i^{ext} \right\rangle + Nk_B T = \langle W \rangle - PV + Nk_B T$$

  i.e. $P=\rho k_B T+\langle W\rangle/V$

46

46

- By considering a pairwise inter-particle interaction: $\vec{f}_{ij} = -\vec{f}_{ji}$

$$\sum_{i=1}^{N} \vec{r}_i \cdot \vec{f}_i = \sum_{i=1}^{N} \sum_{j=1(\neq i)}^{N} \vec{r}_i \cdot \vec{f}_{ij} = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1(\neq i)}^{N} \left(\vec{r}_i \cdot \vec{f}_{ij} + \vec{r}_j \cdot \vec{f}_{ji} \right) =$$

$$= \sum \sum_{i<j=1}^{N} (\vec{r}_i - \vec{r}_j) \cdot \vec{f}_{ij} = -\sum \sum_{i<j=1}^{N} \vec{r}_{ij} \cdot \vec{\nabla} v\left(|\vec{r}_{ij}|\right) = -\sum \sum_{i<j=1}^{N} \vec{r}_{ij} \cdot \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|} \frac{dv(r)}{dr}\bigg|_{r=|\vec{r}_{ij}|} =$$

$$= -\sum \sum_{i<j=1}^{N} |\vec{r}_{ij}| \frac{dv(r)}{dr}\bigg|_{r=|\vec{r}_{ij}|} = -\sum \sum_{i<j=1}^{N} w\left(|\vec{r}_{ij}|\right) \qquad \text{with} \quad w(r) = r\frac{dv(r)}{dr}$$

- Thus we have obtained that

$$P = \rho k_B T - \frac{1}{3V} \left\langle \sum \sum_{i<j=1}^{N} w\left(|\vec{r}_{ij}|\right) \right\rangle = \rho k_B T - \frac{1}{3V} \left\langle \sum \sum_{i<j=1}^{N} |\vec{r}_{ij}| \frac{dv(r)}{dr}\bigg|_{r=|\vec{r}_{ij}|} \right\rangle$$

- In the case of a Lennard-Jones fluid where $v_{LJ}(r)=4\varepsilon[(\sigma/r)^{12}-(\sigma/r)^{6}]$ we have that

$$P = \rho k_B T + \frac{1}{3V} \left\langle \sum \sum_{i<j=1}^{N} 48\varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \frac{1}{2} \left(\frac{\sigma}{r_{ij}} \right)^{6} \right] \right\rangle$$

47

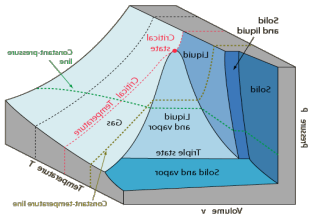
PHYSICAL REVIEW VOLUME 184, NUMBER 1 5 AUGUST 1969

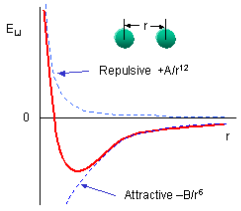
Phase Transitions of the Lennard-Jones System*

Jean-Pierre Hansen and Loup Verlet
Laboratoire de Physique Théorique et Hautes Energies, 91-Orsay, France†
(Received 19 February 1969)

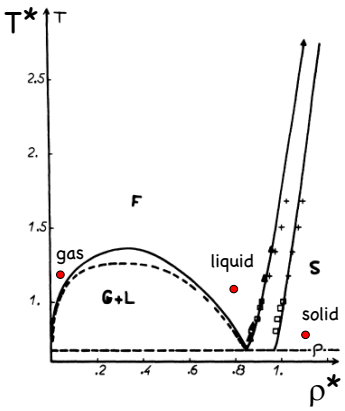
Monte Carlo computations have been performed in order to determine the phase transitions of a system of particles interacting through a Lennard-Jones potential. The fluid-solid transition has been investigated using a method recently introduced by Hoover and Ree. For the liquid-gas transition a method has been devised which forces the system to remain always homogeneous. A comparison is made with experiment in the case of argon. An indirect determination of the phase transition of the hard-sphere gas is made which is essentially in agreement with the results of the more direct calculations.

Termodynamic surface
Note that $\rho \propto 1/V$





$$V_{LJ}(r) = 4\varepsilon \left[\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right]$$



Gas phase: $\rho^*=0.05$ $T^*=1.2$ $r_c=5.0$
Liquid phase: $\rho^*=0.8$ $T^*=1.1$ $r_c=2.5$
Solid phase: $\rho^*=1.1$ $T^*=0.8$ $r_c=2.2$

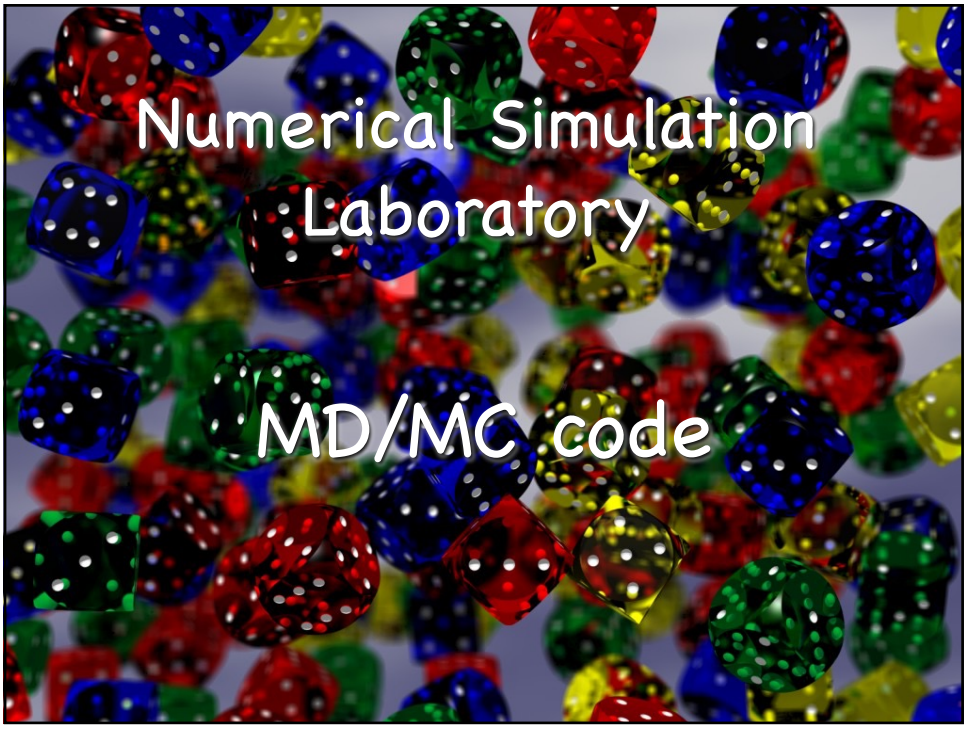
Argon: $T^*=1 \Rightarrow T=120K$ $\rho^*=1 \Rightarrow \rho=24$ atomi/nm^3

48

Lecture 4: Suggested books

- D. Frenkel, *Understanding Molecular Simulation* – AP 2001
- M.P. Allen & D.J. Tildesley, *Computer Simulation of Liquids* – Oxford 2017
- R. Piazza, *Statistical Physics* – Springer 2017
- L. Peliti, *Statistical Mechanics in a Nutshell* – Princeton UP 2011

49



50

Molecular Dynamics/Monte Carlo code: main

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#include <iomanip>
#include "MD_MC.h"

using namespace std;

int main(){
  Input();                              //Inizialization
  int nconf = 1;
  for(int iblk=1; iblk <= nblk; iblk++){ //Data blocking
    Reset(iblk);                        //Reset block averages
    for(int istep=1; istep <= nstep; istep++){
      Move();                           //Move part with Verlet or Metropolis algorithm
      Measure();                        //Mesure properties
      Accumulate();                     //Update block averages
      if(istep%10 == 0){
//        ConfXYZ(nconf);//Write conf in XYZ format; ommented to avoid "filesystem full"!
        nconf += 1;
      }
    }
    Averages(iblk);   //Print results for current block
  }
  ConfFinal(); //Write final configuration

  return 0;
}
```

51

51

MD/MC code: Input()

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#include <iomanip>
#include "MD_MC.h"

using namespace std;

int main(){
  Input();                              //Inizialization
  int nconf = 1;
  for(int iblk=1; iblk <= nblk; iblk++){ //Data blocking
    Reset(iblk);                        //Reset block averages
    for(int istep=1; istep <= nstep; istep++){
      Move();                           //Move part with Verlet or Metropolis algorithm
      Measure();                        //Mesure properties
      Accumulate();                     //Update block averages
      if(istep%10 == 0){
//        ConfXYZ(nconf);//Write conf in XYZ format; ommented to avoid "filesystem full"!
        nconf += 1;
      }
    }
    Averages(iblk);   //Print results for current block
  }
  ConfFinal(); //Write final configuration

  return 0;
}
```

52

52

MD/MC code: Input() 1.


```
void Input(void){
  ifstream ReadInput, ReadConf, ReadVelocity, Primes, Seed;

  cout << "Classic Lennard-Jones fluid        " << endl;
  cout << "MD(NVE) / MC(NVT) simulation       " << endl << endl;
  cout << "Interatomic potential v(r) = 4 * [(1/r)^12 - (1/r)^6]" << endl << endl;
  cout << "Boltzmann weight exp(- beta * sum_{i<j} v(r_ij) ), beta = 1/T " << endl << endl;
  cout << "The program uses Lennard-Jones units " << endl;

  int p1, p2;                //Read seed for random numbers
  Primes.open("Primes");
  Primes >> p1 >> p2 ;
  Primes.close();

  ReadInput.open("input.in");//Read input informations

  ReadInput >> iNVET;        // 0 = MD (NVE); 1 = MC (NVT)
  ReadInput >> restart;      // 0 = NO restart; 1 = restart

  if(restart) Seed.open("seed.out");
  else Seed.open("seed.in");
  Seed >> seed[0] >> seed[1] >> seed[2] >> seed[3];
  rnd.SetRandom(seed,p1,p2);
  Seed.close();

  ReadInput >> temp;
  beta = 1.0/temp;
  cout << "Temperature = " << temp << endl;

  ReadInput >> npart;
  cout << "Number of particles = " << npart << endl;
```

… continues in the next slide …

Input.dat

```
0
0
1.1
108
0.8
2.5
0.0005
20
2000
```

```
ReadInput >> iNVET; 0=MD(NVE)
ReadInput >> restart;
ReadInput >> temp;
ReadInput >> npart;
ReadInput >> rho;
ReadInput >> rcut;
ReadInput >> delta;
ReadInput >> nblk;
ReadInput >> nstep;
```


MD/MC code: Input() 2.


```
  ReadInput >> rho;
  cout << "Density of particles = " << rho << endl;
  vol = (double)npart/rho;
  box = pow(vol,1.0/3.0);
  cout << "Volume of the simulation box = " << vol << endl;
  cout << "Edge of the simulation box = " << box << endl;

  ReadInput >> rcut;
  cout << "Cutoff of the interatomic potential = " << rcut << endl << endl;

  ReadInput >> delta;

  ReadInput >> nblk;

  ReadInput >> nstep;

  cout << "Moves parameter = " << delta << endl;
  cout << "Number of blocks = " << nblk << endl;
  cout << "Number of steps in one block = " << nstep
                                            << endl << endl;
  ReadInput.close();

//Prepare arrays for measurements
  iv = 0; //Potential energy
  it = 1; //Temperature
  ik = 2; //Kinetic energy
  ie = 3; //Total energy
  n_props = 4; //Number of observables
```

$$dt^* = dt\sqrt{\frac{\varepsilon}{m\sigma^2}}$$

… continues in the next slide …

Input.dat

```
0
0
1.1
108
0.8
2.5
0.0005
20
2000
```

$dt = 0{,}109\,fs$

```
ReadInput >> iNVET; 0=MD(NVE)
ReadInput >> restart;
ReadInput >> temp;
ReadInput >> npart;
ReadInput >> rho;
ReadInput >> rcut;
ReadInput >> delta;
ReadInput >> nblk;
ReadInput >> nstep;
```

MD/MC code: Input() 3.

55

```
cout << "Read initial configuration" << endl << endl;
if(restart){
  ReadConf.open("config.out");
  ReadVelocity.open("velocity.out");
  for (int i=0; i<npart; ++i) ReadVelocity >> vx[i] >> vy[i] >> vz[i]; //Read old velocities
} else {
  ReadConf.open("config.in");
  cout << "Prepare velocities with center of mass velocity equal to zero " << endl;
  double sumv[3] = {0.0, 0.0, 0.0};
  for (int i=0; i<npart; ++i){
    vx[i] = rnd.Gauss(0.,sqrt(temp)); //Maxwell-Boltzmann velocity distribution
    vy[i] = rnd.Gauss(0.,sqrt(temp));
    vz[i] = rnd.Gauss(0.,sqrt(temp));
    sumv[0] += vx[i];                 //Compute drift velocity
    sumv[1] += vy[i];
    sumv[2] += vz[i];
  }
  for (int idim=0; idim<3; ++idim) sumv[idim] /= (double)npart;
  double sumv2 = 0.0, fs;
  for (int i=0; i<npart; ++i){
    vx[i] = vx[i] - sumv[0];          //Subtract drift velocity per particle
    vy[i] = vy[i] - sumv[1];
    vz[i] = vz[i] - sumv[2];
    sumv2 += vx[i]*vx[i] + vy[i]*vy[i] + vz[i]*vz[i];
  }
  sumv2 /= (double)npart;
  fs = sqrt(3 * temp / sumv2);        //fs = velocity scale factor
  cout << "velocity scale factor: " << fs << endl << endl;
  for (int i=0; i<npart; ++i){
    vx[i] *= fs;                      //Scale velocities
    vy[i] *= fs;
    vz[i] *= fs;
  }
}
```

$$v^* = \sqrt{\frac{2\langle K \rangle}{\varepsilon}} = \sqrt{\frac{3k_B T}{2}\frac{2}{\varepsilon}} = \sqrt{3T^*}$$

… continues in the next slide …

55

MD/MC code: Input() 4.

56

```
for (int i=0; i<npart; ++i){
  ReadConf >> x[i] >> y[i] >> z[i]; //Read configuration (side units)
  x[i] = Pbc( x[i] * box );         //Scale configuration (reduced unts)
  y[i] = Pbc( y[i] * box );
  z[i] = Pbc( z[i] * box );
}
ReadConf.close();

for (int i=0; i<npart; ++i){
  if(iNVET){
    xold[i] = x[i];
    yold[i] = y[i];
    zold[i] = z[i];
  } else {.        //MD simulation
    xold[i] = Pbc(x[i] - vx[i] * delta); //Compute old configuration from velocities
    yold[i] = Pbc(y[i] - vy[i] * delta);
    zold[i] = Pbc(z[i] - vz[i] * delta);
  }
}

//Evaluate properties of the initial configuration
  Measure();

//Print initial values for measured properties
  cout << "Initial potential energy = " << walker[iv]/(double)npart << endl;
  cout << "Initial temperature      = " << walker[it] << endl;
  cout << "Initial kinetic energy   = " << walker[ik]/(double)npart << endl;
  cout << "Initial total energy     = " << walker[ie]/(double)npart << endl;

  return;
}
```

MD/MC code: Pbc

```
double Pbc(double r){
//Algorithm for periodic boundary conditions with side L=box
    return r - box * rint(r/box);
}
```

56

MD/MC code: Reset(iblk)

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#include <iomanip>
#include "MD_MC.h"

using namespace std;

int main(){
  Input();
  int nconf = 1;
  for(int iblk=1; iblk <=
    Reset(iblk);
    for(int istep=1; iste
      Move();                                          ropolis algorithm
      Measure();
      Accumulate();
      if(istep%10 == 0){
//        ConfXYZ(nconf);                               "filesystem full"!
        nconf += 1;
      }
    }
    Averages(iblk);   //P
  }
  ConfFinal(); //Write final configuration

  return 0;
}
```

```
void Reset(int iblk) //Reset block averages
{
   if(iblk == 1){
       for(int i=0; i<n_props; ++i){
           glob_av[i] = 0;
           glob_av2[i] = 0;
       }
   }

   for(int i=0; i<n_props; ++i){
     blk_av[i] = 0;
   }
   blk_norm = 0;
   attempted = 0;
   accepted = 0;
}
```

57

57

MD/MC code: Move()

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#include <iomanip>
#include "MD_MC.h"

using namespace std;

int main(){
  Input();                                //Inizialization
  int nconf = 1;
  for(int iblk=1; iblk <= nblk; iblk++){ //Data blocking
    Reset(iblk);                          //Reset block averages
    for(int istep=1; istep <= nstep; istep++){
      Move();                             //Move part with Verlet or Metropolis algorithm
      Measure();                          //Mesure properties
      Accumulate();                       //Update block averages
      if(istep%10 == 0){
//        ConfXYZ(nconf);//Write conf in XYZ format; ommented to avoid "filesystem full"!
        nconf += 1;
      }
    }
    Averages(iblk);   //Print results for current block
  }
  ConfFinal(); //Write final configuration

  return 0;
}
```

58

58

59

MD/MC code: Move( )

```
void Move(){
  int o;
  double p, energy_old, energy_new;
  double xnew, ynew, znew;
  if(iNVET){ // Monte Carlo (NVT) move
    //...
  } else {                          //Molecular Dynamics (NVE) move
    double fx[m_part], fy[m_part], fz[m_part];
    for(int i=0; i<npart; ++i){   //Force acting on particle i
      fx[i] = Force(i,0);
      fy[i] = Force(i,1);
      fz[i] = Force(i,2);
    }
    for(int i=0; i<npart; ++i){   //Verlet integration scheme
      xnew = Pbc( 2.0 * x[i] - xold[i] + fx[i] * pow(delta,2) );
      ynew = Pbc( 2.0 * y[i] - yold[i] + fy[i] * pow(delta,2) );
      znew = Pbc( 2.0 * z[i] - zold[i] + fz[i] * pow(delta,2) );
      vx[i] = Pbc(xnew - xold[i])/(2.0 * delta); //Compute velocities
      vy[i] = Pbc(ynew - yold[i])/(2.0 * delta);
      vz[i] = Pbc(znew - zold[i])/(2.0 * delta);
      xold[i] = x[i];
      yold[i] = y[i];
      zold[i] = z[i];
      x[i] = xnew;
      y[i] = ynew;
      z[i] = znew;
      accepted = accepted + 1.0;  //Update acceptance measure
      attempted = attempted + 1.0;
    }
  }
  return;
}
```

$$\vec{r}(t+dt) \cong 2\vec{r}(t) - \vec{r}(t-dt) + \frac{dt^2}{m}\vec{F}(t)$$

MD/MC code: Pbc

```
double Pbc(double r){
//Algorithm for periodic boundary conditions with side L=box
    return r - box * rint(r/box);
}
```

59

60

MD/MC code: Move( )

```
void Move(){
  int o;
  double p, energy_old, energy_new;
  double xnew, ynew, znew;
  if(iNVET){ // Monte Carlo (NVT) move
    //...
  } else {                          //Molecular Dynamics (NVE) move
    double fx[m_part], fy[m_part], fz[m_part];
    for(int i=0; i<npart; ++i){   //Force acting on particle i
      fx[i] = Force(i,0);
      fy[i] = Force(i,1);
      fz[i] = Force(i,2);
    }
    for(int i=0; i<npart; ++i){   //Verlet integration scheme
      xnew = Pbc( 2.0 * x[i] - xold[i] + fx[i] * pow(delta,2) );
      ynew = Pbc( 2.0 * y[i] - yold[i] + fy[i] * pow(delta,2) );
      znew = Pbc( 2.0 * z[i] - zold[i] + fz[i] * pow(delta,2) );
      vx[i] = Pbc(xnew - xold[i])/(2.0 * delta); //Compute velocities
      vy[i] = Pbc(ynew - yold[i])/(2.0 * delta);
      vz[i] = Pbc(znew - zold[i])/(2.0 * delta);
      xold[i] = x[i];
      yold[i] = y[i];
      zold[i] = z[i];
      x[i] = xnew;
      y[i] = ynew;
      z[i] = znew;
      accepted = accepted + 1.0;  //Update acceptance measure
      attempted = attempted + 1.0;
    }
  }
  return;
}
```

60

MD/MC code: Force(i,0/1/2)

```
double Force(int ip, int idir){ //Compute forces as -Grad_ip V(r)
  double f=0.0;
  double dvec[3], dr;


  for (int i=0; i<npart; ++i){
    if(i != ip){
      dvec[0] = Pbc( x[ip] - x[i] );  // distance ip-i in pbc
      dvec[1] = Pbc( y[ip] - y[i] );
      dvec[2] = Pbc( z[ip] - z[i] );


      dr = dvec[0]*dvec[0] + dvec[1]*dvec[1] + dvec[2]*dvec[2];
      dr = sqrt(dr);


      if(dr < rcut){
        f += dvec[idir] * (48.0/pow(dr,14) - 24.0/pow(dr,8)); // -Grad_ip V(r)
      }
    }
  }

  return f;
}
```

$$\vec{F}_{ij}^{LJ}(r) = -\vec{\nabla}_i 4\varepsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] = -4\varepsilon\left[-12\left(\frac{\sigma}{r_{ij}}\right)^{12}\frac{\vec{r}_i - \vec{r}_j}{r_{ij}^2} + 6\left(\frac{\sigma}{r_{ij}}\right)^{6}\frac{\vec{r}_i - \vec{r}_j}{r_{ij}^2}\right] =$$

$$= 48\varepsilon\left(\frac{\sigma}{r_{ij}}\right)^{14}\frac{\vec{r}_i - \vec{r}_j}{\sigma^2} - 24\varepsilon\left(\frac{\sigma}{r_{ij}}\right)^{8}\frac{\vec{r}_i - \vec{r}_j}{\sigma^2}$$

$$\vec{F}_{ij}^{*}(r) = \vec{F}_{ij}^{LJ}(r)\frac{\sigma}{\varepsilon}$$

61

61

MD/MC code: Measure()

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#include <iomanip>
#include "MD_MC.h"

using namespace std;

int main(){
  Input();                              //Inizialization
  int nconf = 1;
  for(int iblk=1; iblk <= nblk; iblk++){ //Data blocking
    Reset(iblk);                        //Reset block averages
    for(int istep=1; istep <= nstep; istep++){
      Move();                           //Move part with Verlet or Metropolis algorithm
      Measure();                        //Mesure properties
      Accumulate();                     //Update block averages
      if(istep%10 == 0){
//        ConfXYZ(nconf);//Write conf in XYZ format; ommented to avoid "filesystem full"!
        nconf += 1;
      }
    }
    Averages(iblk);   //Print results for current block
  }
  ConfFinal(); //Write final configuration

  return 0;
}
```

62

62

MD/MC code: Measure()

```
void Measure(){ //Properties measurement
  double v = 0.0, kin=0.0;
  double vij;
  double dx, dy, dz, dr;

//cycle over pairs of particles
  for (int i=0; i<npart-1; ++i){
    for (int j=i+1; j<npart; ++j){
// distance i-j in pbc = Minimum Image Convention
      dx = Pbc(x[i] - x[j]);
      dy = Pbc(y[i] - y[j]);
      dz = Pbc(z[i] - z[j]);

      dr = dx*dx + dy*dy + dz*dz;
      dr = sqrt(dr);

      if(dr < rcut){//Potential cut-off
        vij = 1.0/pow(dr,12) - 1.0/pow(dr,6);
        v += vij;
      }
    }
  }

//Measure Kinetic energy
  for (int i=0; i<npart; ++i) kin += 0.5 * (vx[i]*vx[i] + vy[i]*vy[i] + vz[i]*vz[i]);

  walker[iv] = 4.0 * v; // Potential energy
  walker[ik] = kin; // Kinetic energy
  walker[it] = (2.0 / 3.0) * kin/(double)npart; // Temperature
  walker[ie] = 4.0 * v + kin;  // Total energy;

  return;
}
```

$$v^*_{LJ}(r) = 4\left[\left(\frac{1}{r}\right)^{12} - \left(\frac{1}{r}\right)^{6}\right]$$

$$K^* = \frac{1}{2}v_1^{*2} + \cdots + \frac{1}{2}v_N^{*2}$$

$$T^* = \frac{2}{3}\left\langle K^* \right\rangle$$

63

63

MD/MC code: Accumulate()

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#include <iomanip>
#include "MD_MC.h"

using namespace std;

int main(){
  Input();
  int nconf = 1;
  for(int iblk=1; iblk
    Reset(iblk);
    for(int istep=1; is
      Move();                                   tropolis algorithm
      Measure();
      Accumulate();
      if(istep%10 == 0){
//        ConfXYZ(nconf);//Write conf in XYZ format; ommented to avoid "filesystem full"!
        nconf += 1;
      }
    }
    Averages(iblk);   //Print results for current block
  }
  ConfFinal(); //Write final configuration

  return 0;
}
```

```
void Accumulate(void) //Update block averages
{

   for(int i=0; i<n_props; ++i)
   {
     blk_av[i] = blk_av[i] + walker[i];
   }
   blk_norm = blk_norm + 1.0;
}
```

64

64

MD/MC code: ConfXYZ(nconf)

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#i
#i

us

ir
```

```
void ConfXYZ(int nconf){ //Write configuration in .xyz format
  ofstream WriteXYZ;

  WriteXYZ.open("frames/config_" + to_string(nconf) + ".xyz");
  WriteXYZ << npart << endl;
  WriteXYZ << "This is only a comment!" << endl;
  for (int i=0; i<npart; ++i){
    WriteXYZ << "LJ  " << Pbc(x[i]) << "   " <<  Pbc(y[i]) << "   " << Pbc(z[i]) << endl;
  }
  WriteXYZ.close();
}
```

```
      Accumulate();                    //Update block averages
      if(istep%10 == 0){
//        ConfXYZ(nconf);//Write conf in XYZ format; ommented to avoid "filesystem full"!
        nconf += 1;
      }
    }
    Averages(iblk);   //Print results for current block
  }
  ConfFinal(); //Write final configuration

  return 0;
}
```

65

65

MD/MC code: Averages(iblk)

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#include <iomanip>
#include "MD_MC.h"

using namespace std;

int main(){
  Input();                             //Inizialization
  int nconf = 1;
  for(int iblk=1; iblk <= nblk; iblk++){ //Data blocking
    Reset(iblk);                       //Reset block averages
    for(int istep=1; istep <= nstep; istep++){
      Move();                          //Move part with Verlet or Metropolis algorithm
      Measure();                       //Mesure properties
      Accumulate();                    //Update block averages
      if(istep%10 == 0){
//        ConfXYZ(nconf);//Write conf in XYZ format; ommented to avoid "filesystem full"!
        nconf += 1;
      }
    }
    Averages(iblk);   //Print results for current block
  }
  ConfFinal(); //Write final configuration

  return 0;
}
```

66

66

MD/MC code: Averages(iblk) 1.

```
void Averages(int iblk) { //Print results for current block

   ofstream Epot, Ekin, Etot, Temp;
   const int wd=12;

    cout << "Block number " << iblk << endl;
    cout << "Acceptance rate " << accepted/attempted << endl << endl;

    Epot.open("output_epot.dat",ios::app);
    Ekin.open("output_ekin.dat",ios::app);
    Temp.open("output_temp.dat",ios::app);
    Etot.open("output_etot.dat",ios::app);

    stima_pot = blk_av[iv]/blk_norm/(double)npart; //Potential energy
    glob_av[iv] += stima_pot;
    glob_av2[iv] += stima_pot*stima_pot;
    err_pot=Error(glob_av[iv],glob_av2[iv],iblk);

    stima_kin = blk_av[ik]/blk_norm/(double)npart; //Kinetic energy
    glob_av[ik] += stima_kin;
    glob_av2[ik] += stima_kin*stima_kin;
    err_kin=Error(glob_av[ik],glob_av2[ik],iblk);
```

MD/MC code: Error

```
double Error(double sum, double sum2, int iblk){
    return sqrt(fabs(sum2/(double)iblk - pow(sum/(double)iblk,2))/(double)iblk);
}
```

67

... continues in the next slide ...

67

MD/MC code: Averages(iblk) 2.

```
    stima_etot = blk_av[ie]/blk_norm/(double)npart; //Total energy
    glob_av[ie] += stima_etot;
    glob_av2[ie] += stima_etot*stima_etot;
    err_etot=Error(glob_av[ie],glob_av2[ie],iblk);

    stima_temp = blk_av[it]/blk_norm; //Temperature
    glob_av[it] += stima_temp;
    glob_av2[it] += stima_temp*stima_temp;
    err_temp=Error(glob_av[it],glob_av2[it],iblk);

//Potential energy per particle
    Epot << setw(wd) << iblk <<  setw(wd) << stima_pot << setw(wd) <<
glob_av[iv]/(double)iblk << setw(wd) << err_pot << endl;
//Kinetic energy
    Ekin << setw(wd) << iblk <<  setw(wd) << stima_kin << setw(wd) <<
glob_av[ik]/(double)iblk << setw(wd) << err_kin << endl;
//Total energy
    Etot << setw(wd) << iblk <<  setw(wd) << stima_etot << setw(wd) <<
glob_av[ie]/(double)iblk << setw(wd) << err_etot << endl;
//Temperature
    Temp << setw(wd) << iblk <<  setw(wd) << stima_temp << setw(wd) <<
glob_av[it]/(double)iblk << setw(wd) << err_temp << endl;

    cout << "----------------------------" << endl << endl;

    Epot.close();
    Ekin.close();
    Etot.close();
    Temp.close();
}
```

68

68

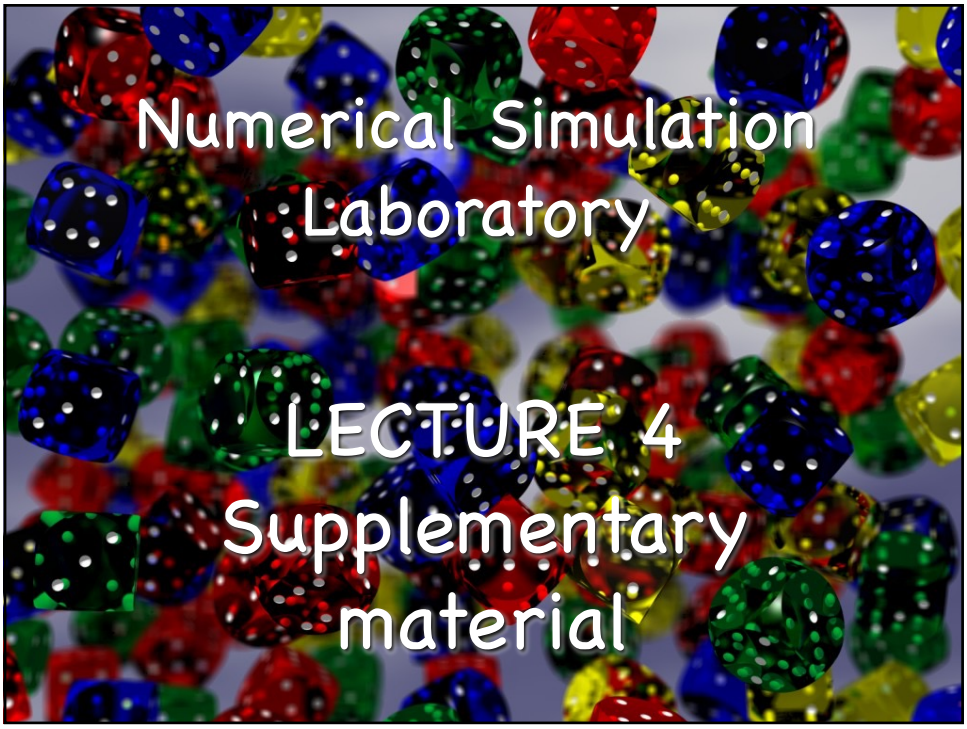
MD/MC code: ConfFinal()

```
#include <iostream>
#include <fstream>
#include <ostream>
#include <cmath>
#include
#include

using na

int main
  Input(
  int no
  for(in
    Rese
    for(
      Mo
      Me
      Ac
      if
//
    }
  }
  Averages(iblk);   //Print results for current block
 }
 ConfFinal(); //Write final configuration

 return 0;
}
```

```
void ConfFinal(void)
{
  ofstream WriteConf, WriteVelocity, WriteSeed;

  cout << "Print final configuration to file config.out" << endl << endl;
  WriteConf.open("config.out");
  WriteVelocity.open("velocity.out");

  for (int i=0; i<npart; ++i){
    WriteConf << x[i]/box << "   " <<  y[i]/box << "   " << z[i]/box << endl;
    WriteVelocity << vx[i] << "   " <<  vy[i] << "   " << vz[i] << endl;
  }

  WriteConf.close();
  WriteVelocity.close();

  rnd.SaveSeed();
}
```

69

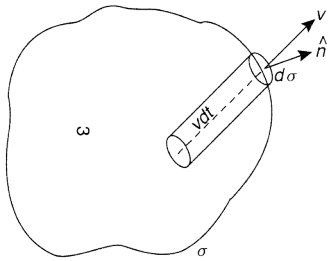
69



70

The "hydrodynamics" of the representative points in Γ

- Since the total number of systems in an ensemble is conserved, the number of representative points leaving any volume in Γ space per second must be equal to the rate of decrease of the number of representative points in the same volume.
- Consider an arbitrary "volume" ω in the relevant region of the phase space and let the "surface" enclosing this volume be denoted by σ
- Then, the rate at which the number of representative points in this volume changes with time is written as



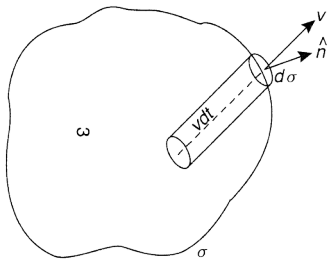
$$\frac{\partial}{\partial t}\int_{\omega}\rho(q,p,t)\,d^{3N}q\,d^{3N}p$$

71

- If we denote by **v** the 6N-dimensional vector, whose components are

$$\vec{v}\equiv\left\{\dot{q}_1,\cdots,\dot{q}_{3N};\dot{p}_1,\cdots,\dot{p}_{3N}\right\}\quad,$$

as the velocity vector of the representative points in the region of the surface element dσ ⇒ the net rate at which the representative points "flow" in/out of ω (across the bounding surface σ) is given by



$$\int_{\sigma}\rho(q,p,t)\,\vec{v}\cdot\hat{\vec{n}}\,d\sigma$$

- n is the (outward) unit vector normal to this element
- By the divergence theorem, this can be written as

$$\int_{\omega}\vec{\nabla}\cdot\left[\rho(q,p,t)\,\vec{v}\right]d^{3N}q\,d^{3N}p$$

- The operation of divergence here means the following:

$$\vec{\nabla}\cdot\left[\rho(q,p,t)\,\vec{v}\right]=\sum_{i=1}^{3N}\left\{\frac{\partial}{\partial q_i}\left[\rho(q,p,t)\,\dot{q}_i\right]+\frac{\partial}{\partial p_i}\left[\rho(q,p,t)\,\dot{p}_i\right]\right\}$$

72

- In view of the fact that there are no "sources" or "sinks" in the phase space and hence the total number of representative points remains conserved, we have

$$\frac{\partial}{\partial t}\int_{\omega}\rho(q,p,t)\,d^{3N}q\,d^{3N}p=-\int_{\omega}\vec{\nabla}\cdot\left[\rho(q,p,t)\,\vec{v}\right]d^{3N}q\,d^{3N}p$$

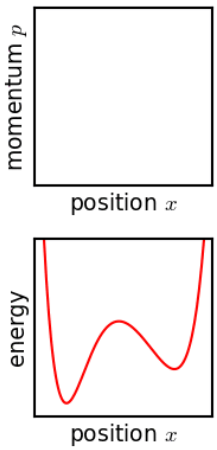
- That is

$$\int_{\omega}\left\{\frac{\partial\rho(q,p,t)}{\partial t}+\vec{\nabla}\cdot\left[\rho(q,p,t)\,\vec{v}\right]\right\}d^{3N}q\,d^{3N}p=0$$

- Now, the necessary and sufficient condition that this integral vanish for all arbitrary volumes ω is that the integrand itself vanish everywhere in the relevant region of the phase space. Thus, we must have

$$\frac{\partial\rho(q,p,t)}{\partial t}+\vec{\nabla}\cdot\left[\rho(q,p,t)\,\vec{v}\right]=0$$

which is the equation of continuity for the swarm of the representative points.



Evolution of an ensemble of classical systems in phase space (top). Each system consists of one massive particle in a 1D well. Liouville's equations describe the flow of the whole distribution

73

73

Liouville's theorem

- We obtain

$$\frac{\partial\rho(q,p,t)}{\partial t}+\sum_{i=1}^{3N}\left[\frac{\partial\rho(q,p,t)}{\partial q_i}\dot{q}_i+\frac{\partial\rho(q,p,t)}{\partial p_i}\dot{p}_i\right]+\rho(q,p,t)\sum_{i=1}^{3N}\left(\frac{\partial\dot{q}_i}{\partial q_i}+\frac{\partial\dot{p}_i}{\partial p_i}\right)=0$$

- The last group of terms vanishes identically because, by the equations of motion, we have, for all i,

$$\frac{\partial\dot{q}_i}{\partial q_i}=\frac{\partial^2 H}{\partial q_i\partial p_i}=\frac{\partial^2 H}{\partial p_i\partial q_i}=-\frac{\partial\dot{p}_i}{\partial p_i}$$

- The remaining terms may be combined to form the "total" time derivative of p, with the result that

Poisson's brackets

$$\frac{d\rho}{dt}=\frac{\partial\rho}{\partial t}+\{\rho,H\}=\frac{\partial\rho}{\partial t}+\sum_i\left(\frac{\partial\rho}{\partial q_i}\frac{\partial H}{\partial p_i}-\frac{\partial\rho}{\partial p_i}\frac{\partial H}{\partial q_i}\right)=0$$

- This is the so-called Liouville 's theorem (1838): the "local" density of the representative points stays constant in time.

74

74

Other algorithms

- Let us now briefly look at some alternatives to the Verlet algorithm. The most naive algorithm is based simply on a truncated Taylor expansion of the particle coordinates:

$$\vec{r}(t+\delta t) = \vec{r}(t) + \delta t\,\vec{v}(t) + \frac{1}{2}\delta t^2\,\vec{a}(t) + O(\delta t^3)$$

- If we truncate this expansion beyond the term in δt^2, we obtain the so called Euler algorithm. Although it looks similar to the Verlet algorithm, it is much worse on virtually all counts. In particular, it is not reversible or phase-space volume preserving (Liouville's theorem) and suffers from a (catastrophic) energy drift. The Euler algorithm therefore is not recommended.
- Several algorithms are equivalent to the Verlet scheme. The simplest among these is the so-called Leap Frog algorithm. This algorithm evaluates the velocities at half-integer time steps and uses these velocities to compute the new positions. To derive the Leap Frog algorithm from the Verlet scheme, we start by defining the velocities at half-integer time steps as follows:

$$\vec{v}(t-\delta t/2) \equiv \frac{\vec{r}(t)-\vec{r}(t-\delta t)}{\delta t} \quad \text{and} \quad \vec{v}(t+\delta t/2) \equiv \frac{\vec{r}(t+\delta t)-\vec{r}(t)}{\delta t}$$

75

75

- From the latter equation we immediately obtain an expression for the new positions, based on the old positions and velocities:

$$\vec{r}(t+\delta t) = \vec{r}(t) + \delta t\,\vec{v}(t+\delta t/2)$$

- From the Verlet algorithm, we get the following expression for the update of the velocities:

$$\vec{v}(t+\delta t/2) - \vec{v}(t-\delta t/2) = \frac{\vec{r}(t+\delta t) - 2\vec{r}(t) + \vec{r}(t-\delta t)}{\delta t} = \delta t\,\vec{a}(t)$$

$$\Rightarrow \quad \vec{v}(t+\delta t/2) = \vec{v}(t-\delta t/2) + \delta t\,\frac{\vec{f}(t)}{m}$$

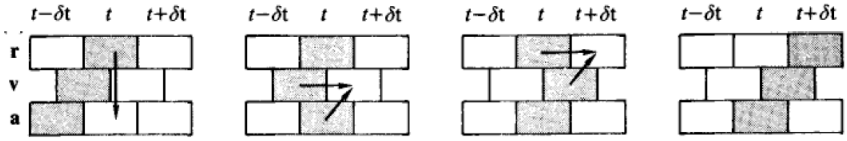
- As the Leap Frog algorithm is derived from the Verlet algorithm, it gives rise to identical trajectories. Note, however, that the velocities are not defined at the same time as the positions. As a consequence, kinetic and potential energy are also not defined at the same time, and hence we cannot directly compute the total energy in the Leap Frog scheme.
- The "actual" velocities may be calculated from

$$\vec{v}(t) = \frac{\vec{v}(t+\delta t/2) + \vec{v}(t-\delta t/2)}{2}$$

76

76

- The overall scheme of this algorithm is illustrated in this figure:



- As the previous equation shows, leap-frog methods still do not handle the velocities in a completely satisfactory manner.
- It is, however, possible to cast the Verlet algorithm in a form that uses positions and velocities computed at equal times. This velocity Verlet algorithm looks like a Taylor expansion for the coordinates:

$$\vec{r}(t+\delta t) = \vec{r}(t) + \delta t\,\vec{v}(t) + \frac{\vec{f}(t)}{2m}\delta t^2$$

- However, the update of the velocities is different from the Euler scheme:

$$\vec{v}(t+\delta t) = \vec{v}(t) + \frac{\vec{f}(t+\delta t) + \vec{f}(t)}{2m}\delta t \qquad \left(\equiv \vec{v}(t+\delta t/2) + \frac{\vec{f}(t+\delta t)}{2m}\delta t\right)$$

- Note that, in this algorithm, we can compute the new velocities only after we have computed the new positions and, from these, the new forces. It is not immediately obvious that this scheme, indeed, is equivalent to the original Verlet algorithm.

77

77

- To show this, we note that

78

$$\vec{r}(t+2\delta t) = \vec{r}(t+\delta t) + \delta t\,\vec{v}(t+\delta t) + \frac{\vec{f}(t+\delta t)}{2m}\delta t^2$$

and the equation for r(t+δt) can be rewritten as

$$\vec{r}(t) = \vec{r}(t+\delta t) - \delta t\,\vec{v}(t) - \frac{\vec{f}(t)}{2m}\delta t^2$$

by addition we get

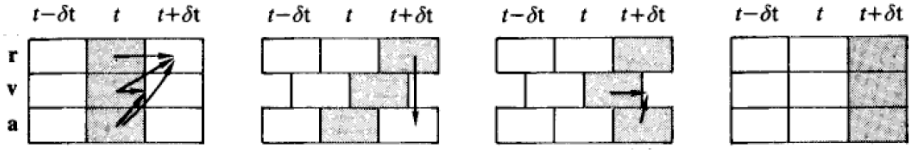
$$\vec{r}(t+2\delta t) + \vec{r}(t) = 2\vec{r}(t+\delta t) + \delta t\left[\vec{v}(t+\delta t) - \vec{v}(t)\right] + \frac{\vec{f}(t+\delta t) - \vec{f}(t)}{2m}\delta t^2 =$$

$$= 2\vec{r}(t+\delta t) + \frac{\vec{f}(t+\delta t) + \vec{f}(t)}{2m}\delta t^2 + \frac{\vec{f}(t+\delta t) - \vec{f}(t)}{2m}\delta t^2 = 2\vec{r}(t+\delta t) + \frac{\vec{f}(t+\delta t)}{m}\delta t^2$$

which, indeed, is the coordinate version of the Verlet algorithm:

$$\vec{r}(t+2\delta t) + \vec{r}(t) = 2\vec{r}(t+\delta t) + \frac{\vec{f}(t+\delta t)}{m}\delta t^2 \Leftrightarrow \vec{r}(t+\delta t) = 2\vec{r}(t) - \vec{r}(t-\delta t) + \frac{\vec{f}(t)}{m}\delta t^2$$

- The overall scheme of this algorithm is illustrated in this figure:



78