Introduction to Simulation - Lectures 17, 18

Molecular Dynamics

Nicolas Hadjiconstantinou

Molecular Dynamics

Molecular dynamics is a technique for computing the equilibrium and non-equilibrium properties of classical* many-body systems.

* The nuclear motion of the constituent particles obeys the laws of classical mechanics (Newton).

References:

1) Computer Simulation of Liquids, M.P. Allen & D.J. Tildesley, Clarendon, Oxford, 1987.
2) Understanding Molecular Simulation: From Algorithms to Applications, D. Frenkel and B. Smit, Academic Press, 1997.
3) Moldy manual

Moldy

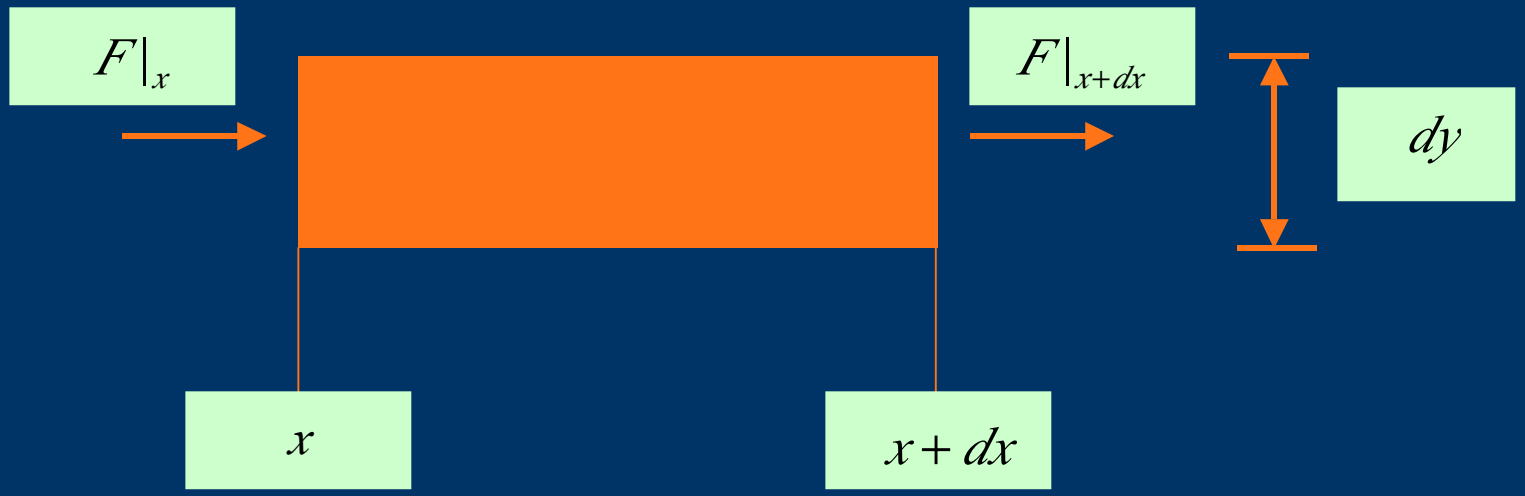
• A free and easy to use molecular dynamics simulation package can be found at the CCP5 program library (http://www.ccp5.ac.uk/librar.shtml), under the name Moldy. At this site a variety of very useful information as well as molecular simulation tools can be found.

• Moldy is easy to use and comes with a very well written manual which can help as a reference. I expect the homework assignments to be completed using Moldy.

Why Do We Need Molecular Dynamics?

Similar to real experiments.

1. Allows us to study and understand material behavior so that we can model it.
2. Tells us what the answer is when we do not have models.
   Example: Diffusion equation



Conservation of mass: $$\frac{d}{dt}\left(ndxdydz\right)=\left(F|_x - F|_{x+dx}\right)dydz$$

n = number density, F = flux

$$\frac{d}{dt}\left(ndxdydz\right) \cong \left(F|_x - \left(F|_x + \frac{\partial F}{\partial x}|_x \, dx + \frac{\partial^2 F}{\partial x^2}\frac{dx^2}{2} \right) \right) dydz$$

$$= -\frac{\partial F}{\partial x} dxdydz - \frac{\partial^2 F}{\partial x^2}\frac{dx}{2} dydzdx$$

in the limit $dx \to 0, \;\Rightarrow \frac{\partial n}{\partial t} + \frac{\partial F}{\partial x} = 0$

- This equation cannot be solved unless a relation between *n* and *F* is provided.
- Experiments or consideration of molecular behavior shows that ***under a variety of conditions***

$$F = -D\frac{\partial n}{\partial x}$$

$$\Rightarrow \frac{\partial n}{\partial t} = D\frac{\partial^2 n}{\partial x^2}$$

diffusion equation!

Breakdown of linear gradient constitutive law

$$F = -D\frac{\partial n}{\partial x}$$

- Large gradients

- Far from equilibrium gaseous flows
  - Shockwaves
  - Small scale flows (high Knudsen number flow)
  - Rarefied flows (low density) (high Knudsen number flow)

High Knudsen number flows (gases)

- Kn is defined as the ratio of the molecular mean-free path to a characteristic lengthscale
- The molecular mean-free path is the average distance traveled by molecules between collisions
- Collisions tend to restore equilibrium
- When Kn « 1 particle motion is diffusive (near equilibrium)
- When Kn » 1 particle motion is ballistic (far from equilibrium)
- $0.1 \leq Kn \leq 10$ physics is transitional (hard to model)

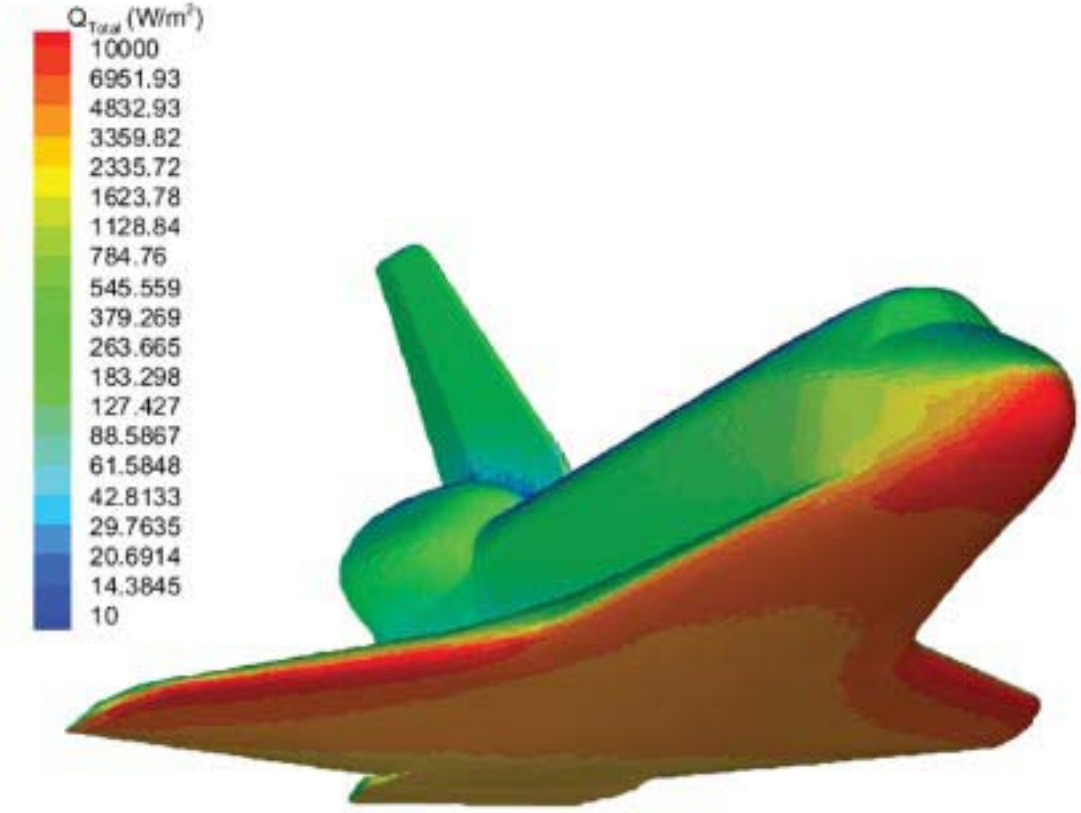
Example: Re-entry vehicle aerobraking maneuver in the upper atmosphere

- In the upper atmosphere density is low (collision rate is low)
- Long mean-free path
- High Knudsen number flows typical

Other high Knudsen number flows

- Small scale flows (mean-free path of air molecules at atmospheric pressure is approximately 60 nanometers)
- Vacuum science (low pressure)

350 kft flowfield
Surface heating



From Dr. M. Gallis of Sandia National Laboratories

Sandia National Laboratories

Brief Intro to Statistical Mechanics

Statistical mechanics provides the theoretical connection between the microscopic description of matter (e.g. positions and velocities of molecules) and the macroscopic description which uses observables such as pressure, density, temperature …

This is achieved through a statistical approach and the concept of an ensemble average. An ensemble is a collection of a large number of identical systems (M) evolving in time under the same *macroscopic* conditions but different microscopic initial conditions.

Let $\rho(\Gamma_i) M$ be the number of such systems in state Γ_i:

Then $\rho(\Gamma_i)$ can be interpreted as the probability of finding an ensemble member in state i.

- Macroscopic properties (observables) are then calculated as weighted averages $\langle A \rangle = \sum_i \rho(\Gamma_i) A(\Gamma_i)$

  or in the continuous limit $\langle A \rangle = \int \rho(\Gamma) A(\Gamma) d\Gamma.$

- One of the fundamental results of statistical mechanics is that the probability of a state of a system with energy E in equilibrium at a fixed temperature T is governed by

$$\rho(\Gamma_E) \propto \exp\left(-\frac{E}{kT}\right)$$

  where k is Boltzmann's constant.

- For non-equilibrium systems solving a problem reduces to the task of calculating $\rho(\Gamma)$.
- Molecular methods are similar to experiments where rather than solving for $\rho(\Gamma)$ we measure $\langle A \rangle$ directly.

$$\langle A \rangle = \sum_i \rho(\Gamma_i) A(\Gamma_i)$$

implies that given an ensemble of systems, any observable $\langle A \rangle$ can be measured by averaging the value of this observable over all systems.

However, in real life we do not use a large number of systems to do experiments. We usually observe one system over some period of time.

This is because we use the ergodic hypothesis:

- Since there is a one-to-one correspondence between the initial conditions of a system and the state at some other time, averaging over a large number of initial conditions is equivalent to averaging over time-evolved states of the system.

- The ergodic hypothesis allows us to convert the averaging from ensemble members to time instances of the same system. THIS IS AN ASSUMPTION THAT SEEMS TO WORK WELL MOST OF THE TIME.

A Simplified MD Program Structure

- Initialize:

  -Read run parameters (initial temperature, number of timesteps, density, number of particles, timestep)

  -Generate or read initial configuration (positions and velocities of particles)

- Loop in timestep increments until $t = t_{final}$

  -Compute forces on particles

  -Integrate equations of motion

  -If $t > t_{equilibrium}$, sample system

- Output results

Equations of Motion

Newton's equations

For $i = 1, \ldots, N$

$$m_i \frac{d^2 \vec{r}_i}{dt^2} = \vec{F}_i = -\frac{\partial}{\partial \vec{r}_i} U\left(\vec{r}_1, \vec{r}_2 \ldots \vec{r}_N\right)$$

$U\left(\vec{r}_1, \vec{r}_2 \ldots \vec{r}_N\right) =$ Potential energy of the system

$$= \sum_i U_1\left(\vec{r}_1\right) + \sum_i \sum_{j>i} U_2\left(\vec{r}_i, \vec{r}_j\right) + \sum_i \sum_{j>i} \sum_{k>j>i} U_3\left(\vec{r}_1, \vec{r}_j, \vec{r}_k\right)$$

$$+ \ldots$$

$U_1\left(\vec{r}_1\right) =$ external field …

$U_2\left(\vec{r}_1, \vec{r}_j\right) =$ pair interaction $= U_2\left(r_{ij}\right), \left(r_{ij}\right) = \left(\vec{r}_i - \vec{r}_j\right)$

$U_3\left(\vec{r}_i, \vec{r}_j, \vec{r}_k\right) =$ three body interaction (expensive to calculate)

For this reason, typically

$$U \cong \sum_i U_1(r_i) + \sum_i \sum_{j>i} U_2^{eff}(r_{ij})$$

where U_2^{eff} includes some of the effects of the three body interactions.

The Lennard-Jones Potential

One of the most simple potentials used is the Lennard-Jones.
Typically used to simulate simple liquids and solids.

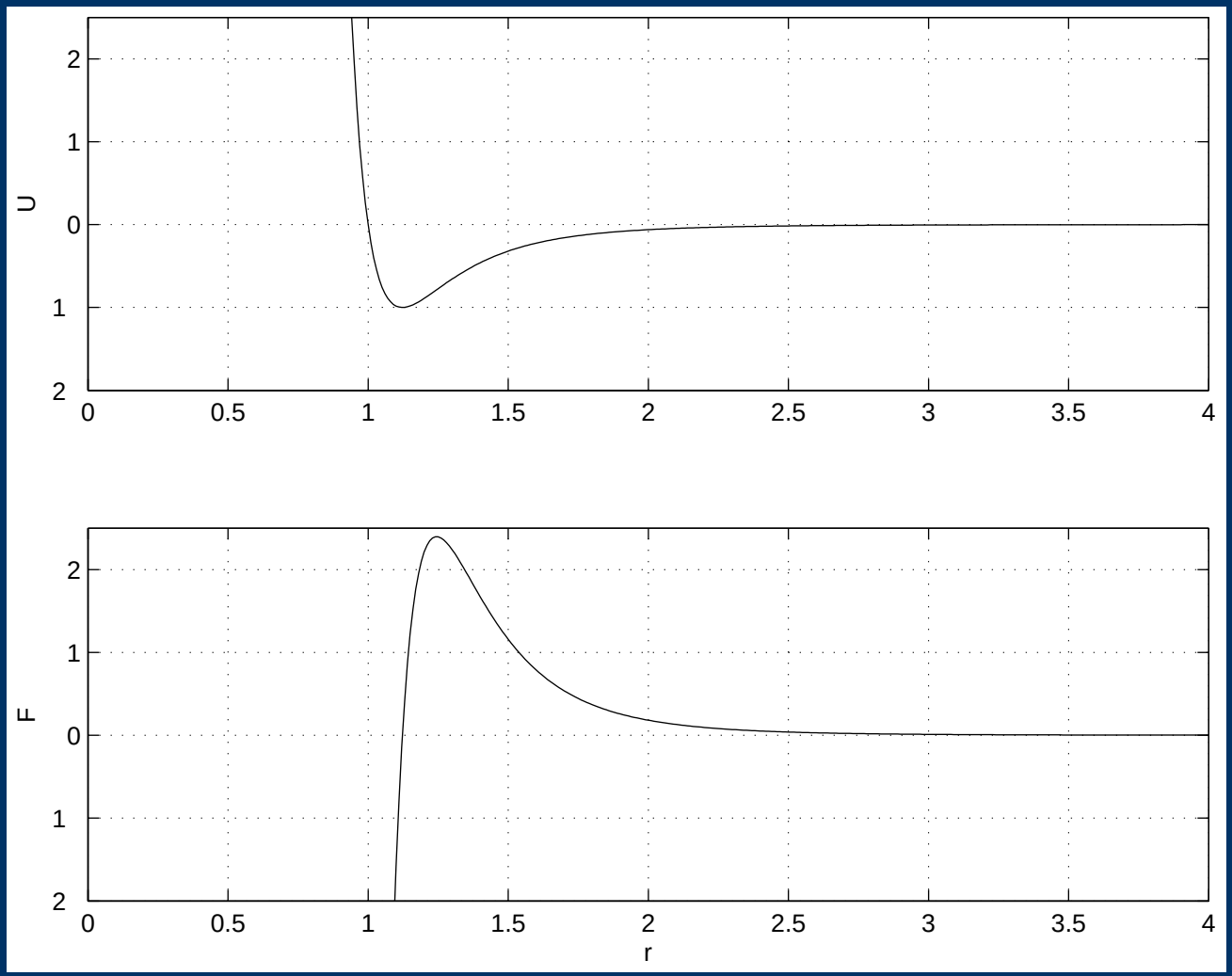
$$U(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

ε is the well depth [energy]

σ is the interaction lengthscale

- Very repulsive for $r < \sigma$
- Potential minimum at $r = \sqrt[6]{2}\,\sigma$
- Weak attraction $\left(\sim 1/r^6\right)$ for $r > 2\sigma$

The Lennard-Jones potential (U) and force (F) as a function of separation (r) ($\varepsilon = \sigma = 1$)



Reduced Units

- What is the evaporation temperature of a Lennard-Jones liquid?
- What is an appropriate timestep for integration of the equations of motion of Lennard-Jones particles?
- What is the density of a liquid/gas made up of Lennard-Jones molecules?

Number density $\rho^* = \rho\sigma^3$

Temperature $T^* = \frac{kT}{\varepsilon}$

Pressure $P^* = \frac{P\sigma^3}{\varepsilon}$

Time $t^* = \sqrt{\frac{\varepsilon}{m\sigma^2}}\, t$

- In these units, numbers have physical significance.
- Results valid for all Lennard-Jones molecules.
- Easier to spot errors: (10^{-32} must be wrong!)

Integration Algorithms

An integration algorithm should

a) Be fast, require little memory

b) Permit the use of a long timestep

c) Duplicate the classical trajectory as closely as possible

d) Satisfy conservation of momentum and energy and be time-reversible

e) Simple and easy to program

Discussion of Integration Algorithms

a) Not very important, by far the most expensive part of simulation is in calculating the forces

b) Very important because for a given total simulation time the longer the timestep the less the number of force evaluation calls

c) Not very important because no numerical algorithm will provide the exact solution for long time (nearby trajectories deviate exponentially in time). Recall that MD is a method for obtaining time averages over "all initial conditions" under prescribed macroscopic constraints. Thus conserving momentum and energy is more important.

d) Very important (see C)

e) Important, no need for complexity when no speed gains are possible.

The Verlet Algorithm

One of the most popular methods for at least the first few decades of MD.

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

$$\vec{a}(t) = \frac{\vec{F}(t)}{m}$$

Derivation: Taylor series expansion of $\vec{r}(t)$ about time t

$$\vec{r}(t+\delta t) = \vec{r}(t) + \delta t\ \vec{V}(t) + \frac{1}{2}\delta t^2 \vec{a}(t) + \ldots$$

$$\vec{r}(t-\delta t) = \vec{r}(t) - \delta t\ \vec{V}(t) + \frac{1}{2}\delta t^2 \vec{a}(t) + \ldots$$

ADVANTAGES

1) Very compact and simple to program
2) Excellent energy conservation properties (helped by time-reversibility)

3) Time reversible $\vec{r}(t+\delta t) \leftrightarrow \vec{r}(t-\delta t)$

4) Local error $\mathcal{O}(\delta t^4)$

DISADVANTAGES

1) Awkward handling of velocities $\vec{V}(t) = \dfrac{\vec{r}(t+\delta t) - \vec{r}(t-\delta t)}{2\delta t}$

a) Need $\vec{r}(t+\delta t)$ solution before getting $\vec{V}(t)$

b) Error for velocities $\mathcal{O}(\delta t^2)$

2) May be sensitive to truncation error because in

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

a small number is added to the difference of two large numbers.

Improvements To The Verlet Algorithm

Beeman Algorithm:

$$\vec{r}(t+\delta t) = \vec{r}(t) + \delta t \vec{V}(t) + \delta t^2 \frac{4\vec{a}(t) - \vec{a}(t-\delta t)}{6}$$

$$\vec{V}(t+\delta t) = \vec{V}(t) + \delta t \frac{2\vec{a}(t+\delta t) + 5\vec{a}(t) - \vec{a}(t-\delta t)}{6}$$

- Coordinates equivalent to Verlet algorithm
- $\vec{V}$ more accurate than Verlet

Predictor Corrector Algorithms

Basic structure:

a) Predict positions, velocities, accelerations at $t + \delta t$.

b) Evaluate accelerations from new positions and velocities (if forces are velocity dependent.

c) Correct the predicted positions, velocities, accelerations using the new accelerations.

d) Go to (a).

Although these methods can be very accurate, the nature of MD simulations is not well suited to them. The reason is that any prediction and correction which does not take into account the motion of the neighbors is unreliable.

The concept of such a method is demonstrated here by the modified Beeman algorithm that handles velocity dependent forces (discussed later).

a) $\vec{r}(t+\delta t) = \vec{r}(t) + \delta t \vec{V}(t) + \frac{\delta t^2}{6}\left[\vec{a}(t) - \vec{a}(t-\delta t)\right]$

b) $\vec{V}^P(t+\delta t) = \vec{V}(t) + \frac{\delta t}{2}\left[3\vec{a}(t) - \vec{a}(t-\delta t)\right]$

c) $\vec{a}(t+\delta t) = \frac{1}{m} F\left\{\vec{r}(t+\delta t), \vec{V}^P(t+\delta t)\right\}$

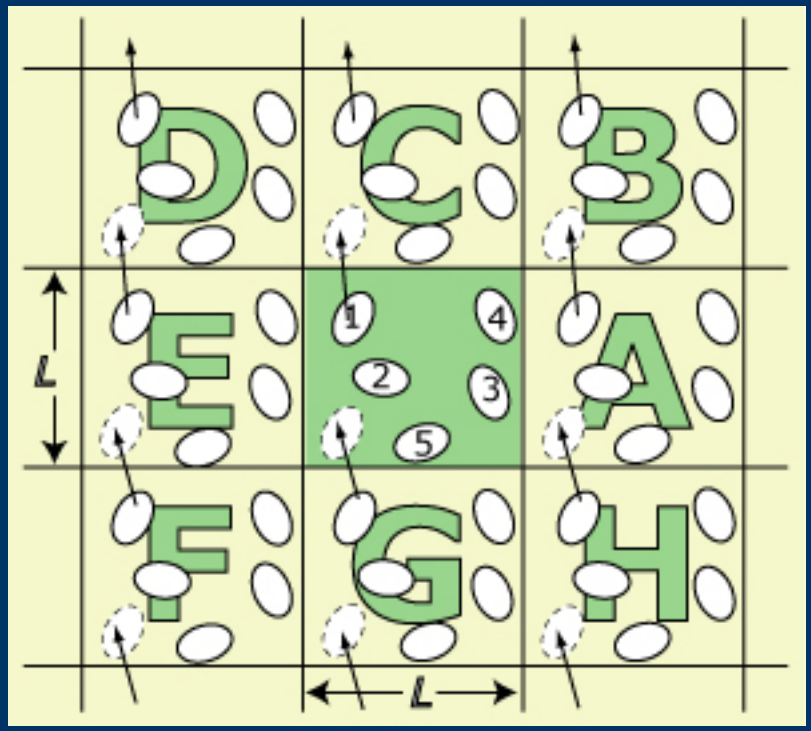
d) $\vec{V}^c(t+\delta t) = \vec{V}(t) + \frac{\delta t}{6}\left[2\vec{a}(t+\delta t) + 5\vec{a}(t) - (t-\delta t)\right]$

e) Replace $\vec{V}^P$ with $\vec{V}^c$ and go to c.

If there are no velocity dependent forces this reduces to the Beeman method discussed above.

Periodic Boundary Conditions

- Periodic boundary conditions are very popular: Reduce surface effects



Adapted from Computer Simulation of Liquids by M.P. Allen & D.J. Tildesley, Oxford Science Publications, 1987.

- Today's computers can easily treat N > 1000 so artifacts from small systems with periodic boundary conditions are limited.
- Equilibrium properties unaffected.
- Long wavelengths not possible.
- In today's implementations, particle interacts with "closest" images of other molecules.

Evaluating Macroscopic Quantities

- Macroscopic quantities (observables) are defined as appropriate averages of microscopic quantities.

$$T = \frac{1}{\frac{3}{2}Nk}\sum_{i=1}^{N}\frac{\vec{P}_i^2}{2m_i}$$

$$\rho = \frac{1}{V}\sum_{i=1}^{N} m_i$$

$$\pi = \frac{1}{V}\left(\sum_i m\vec{V}_i\vec{V}_i + \sum_i\sum_j \vec{r}_{ij}\vec{F}_{ij}\right)$$

$$\vec{u} = \frac{\sum_i \vec{P}_i}{\sum_i m_i}$$

(macroscopic velocity).

- If the system is not in equilibrium, these properties can be defined as a function of space and time by averaging over extents (space, time) over which change is small.

Starting The Simulation Up

- Need initial conditions for positions and velocities of all molecules in the system.
- Typically initial density, temperature, number of particles known.
- Because of the highly non-linear particle interaction starting at completely arbitrary states is almost never successful.

  If particle positions are initialized randomly, with overwhelming probability at least one pair of particles will be very close and lead to energy divergence.

- Velocity degrees of freedom can be safely initialized using the equilibrium distribution

$$P(E) \propto \exp\left(-\frac{E}{kT}\right)$$

because the additive nature of the kinetic energy

$$E_k = \sum_i^N \frac{\vec{P}_i^2}{2m}$$

leads to independent probability distributions for each particle velocity.

- Liquids are typically started in

  - a crystalline solid structure that melts during the "equilibration" part of the simulation

  - a quasi-random structure that avoids particle overlap (see Moldy manual for an example).

Equilibration

Because systems are typically initialized in the incorrect state, potential energy is usually either converted into thermal energy or thermal energy is consumed as the system equilibrates. In order to stop the temperature from drifting, one of the following methods is used:

1) Velocity rescaling $\vec{V}_i' = \sqrt{\frac{T_d}{T}}\vec{V}_i$

where T_d is the desired temperature and

$T = \frac{2}{3Nk}\sum_i^N \frac{\vec{P}_i^2}{2m_i}$ is the instantaneous temperature.

This is the simplest and most crude way of controlling the temperature of the simulation.

2) Thermostat. A thermostat is a method to keep the temperature constant by introducing it as a constraint into the equations of motion. Thermostats will be described under "Constrained Dynamics."

Long Range Forces, Cutoffs, And Neighbor Lists

- Lennard-Jones potential decays as r^{-6} which is reasonably fast.
- However, the number of neighbors interacting with a particle grows as r^{3}.
- Interaction is thus cut off at some distance r_c to limit computational cost.
- Most sensitive quantitites (surface tension) to the long range forces usually calculated with r_c approximately 10σ.
- Typical calculations for hydrodynamics use r_c approximately 2.5σ.
- Electrostatic interactions require special methods (Multiple expansions, Ewald sums) - See Moldy manual

- Although system behavior (properties: equation of state, transport coefficients, latent heat, elastic constants) are affected by r_c, the new ones can be measured, if required.

- The simplest cut-off approach is a truncation

$$U^{tr}(r) = \begin{cases} U(r) & r \le r_c \\ 0 & r > r_c \end{cases}$$

Not favored because $U^{tr}(r)$ is discontinuous. Does not conserve energy.

- This is fixed by the truncated and shifted potential

$$U^{tr-sh}(r) = \begin{cases} U(r) - U(r_c) & r \le r_c \\ 0 & r > r_c \end{cases}$$

- Even with a small cut-off value the calculation cost is proportional to N^2 because of the need to examine all pair separations.

- Cost reduced by “neighbor lists”

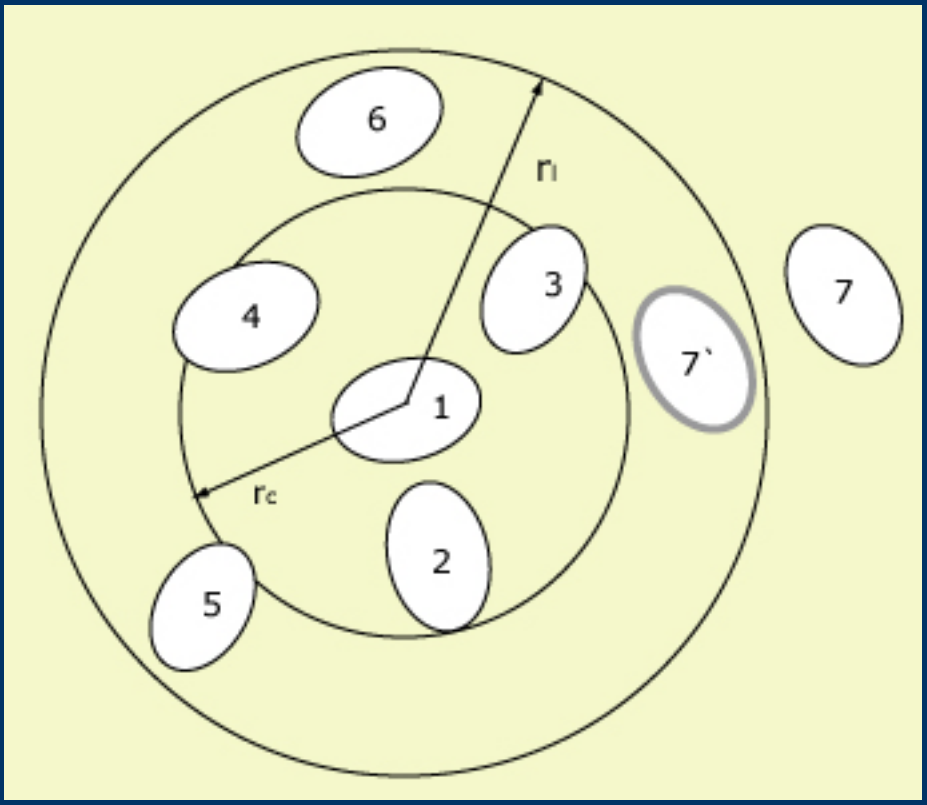
  -Verlet list
  -Cell index method

Verlet Neighbor Lists

- Keep an expanded neighbor list $\left(r_\ell > r_c\right)$
  so that neighbor pairs need not be calculated every timestep

- r_ℓ chosen such that need to test every 10-20 timesteps

- Good for $N \leq 1000$
  (too much storage required)



Adapted from Computer Simulation of Liquids, by M.P. Allen & D.J. Tildesley, Oxford Science Publications, 1987.

Cell-index Method

- Divide simulation into m subcells in each direction (here $m = 5$).
- Search only sub-cells within cut-off (conservative)

Example: If sub-cell size larger than cut-off for cell 13 only cells 7, 8, 9, 12, 13, 14, 17, 18, 19 need to be searched.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

- In two dimensions cost is 4.5 NN_c where

$$N_c = \frac{N}{\overline{m}^2} \quad \text{instead of} \quad \frac{1}{2}N(N-1).$$

- In three dimensions cost is 13.5 NN_c instead of $\frac{1}{2}N(N-1).$

  (With N_c appropriately redefined)

Constraint Methods

- Newtonian molecular dynamics conserves system energy, volume and number of particles.

- Engineering-physical systems typically operate at constant pressure, temperature and exchange mass (i.e. they are open).

- Methods to simulate these have been proposed*.

* These methods are capable of providing the correct statistical description in *equilibrium*. Away from equilibrium there is *no proof* that these methods provide the correct physics. Therefore they need to be used with care, especially the crude ones such as rescaling.

Constant Temperature Simulations

- Newton's equations conserve energy and temperature is a variable.

- In most calculations of practical interest we would like to prescribe temperature (in reality, reservoirs, such as the atmosphere, interact with systems of interest and keep temperature constant).

- Similar considerations apply for pressure. We would like to perform constant pressure calculations with variable system volume.

- Three main types of approaches

  -Velocity rescaling (crude)

- Extended system methods (One or more extra degrees of freedom are added to represent the "reservoir".)

Equations of motion $\frac{d\vec{r}_i}{dt} = \frac{\vec{P}_i}{m}, \quad \frac{d\vec{P}_i}{dt} = \vec{F}_i - f\vec{P}_i$

f is a dynamical variable and is given by $\frac{df}{dt} = \frac{kg}{Q}\left(T - T_d\right)$

g = number of degrees of freedom
Q = reservoir mass (adjustable parameter-controls equilibration dynamics)

This method is known as the Nosé-Hoover thermostat.

- Constraint methods (The equations of motion are constrained to lie on a hyperplane in phase space).

In this case the constraint is $\frac{d}{dt}T \propto \frac{d}{dt}\sum_i P_i^2 = 0$

This leads to the following equations of motion

$$\frac{d\vec{r}_i}{dt} = \frac{\vec{P}_i}{m}, \quad \frac{d\vec{P}_i}{dt} = \vec{F}_i - \lambda\vec{P}_i, \quad \lambda = \frac{\sum_{i=1}^{N} \vec{P}\cdot\vec{F}_i}{\sum_{i=1}^{N} \vec{P}_i^2}$$

- λ is a Lagrange multiplier (not a dynamical variable).
- A discussion of these and constant pressure methods can be found in the Moldy Manual.
- Note the "velocity-dependent forces."

Homework Discussion

- Use (and modify) control file control.argon. You can run short familiarization runs with this file.
- Control.argon calls the argon.in system specification file. Increase the number of particles to ***at least*** 1000 (reduce small system effects and noise).
- Note that density in control.argon is in units equivalent to g/cm^3 (= 1000 Kg/m^3).
- Reduce noise by increasing number of particles and/or averaging interval (**average-interval**).
- Make sure that temperature does not drift significantly from the original temperature. You may want to rescale (**scale-interval**) more often and stop rescaling (**scale-end**) later. **Make sure you are not sampling while rescaling!**
- By setting **rdf-interval = 0** you disable the radial distribution function calculation which makes the output file more readable.

Sample control file

```
#
# Sample control file. Different from the one provided with code.
#
sys-spec-file=argon.in
density=1.335          #density=1335 Kg/m^3
temperature=120        #initial temperature=120K
scale-interval=2       #rescale velocities every 2 timesteps
scale-end=5000         #stop rescaling velocities at timestep 5000
nsteps=25000           #run a total of 25000 timesteps
print-interval=500     #print results every 500 timesteps
roll-interval=500      #keep rolling average results every 500 timesteps
begin-average=5001 #begin averaging at timestep 5001
average-interval=10000 #average for 10000 timesteps
step=0.01              #timestep 0.01ps
subcell=2              #subcell size for linked-cell method
strict-cutoff=1        #calculate forces using "strict" 0(N^2) algorithm
cutoff=10.2            #3 * sigma
begin-rdf=2501 #begin radial distribution function calculation at timestep 2501
rdf-interval=0
rdf-out=2500
```