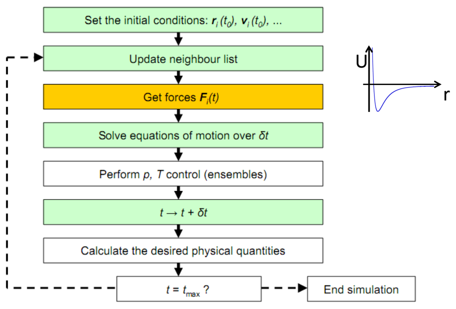
MTLE-4500
Computational Materials Design

Instructor: Liping Huang
huangl5@rpi.edu, MRC 202, x2174
Office hours: T F 11:20 – 12:00 or by appointment
TA: Siddharth Sundararaman (sunds@rpi.edu)
Office hours: M/Th 4-5pm in MRC 208A

Lecture 9: Molecular Dynamics Simulation: Integrators

Flow of Molecular Dynamics Simulation


Set the initial conditions: $\mathbf{r}_i(t_0)$, $\mathbf{v}_i(t_0)$, ...
Update neighbour list
Get forces $\mathbf{F}_i(t)$
Solve equations of motion over δt
Perform p, T control (ensembles)
$t \rightarrow t + \delta t$
Calculate the desired physical quantities
$t = t_{max}$?
End simulation
U
r

A computational experiment

- Initialize: select positions and velocities
➢Second order differential equations: boundary conditions require initial positions and initial velocities
➢Initial positions: reasonably compatible with the structure to be studied. Avoid overlap, short distances. (A simple way is to put them on a regular lattice!)
➢Velocities: assign velocities according to Maxwell-Boltzmann distribution

- Integrate: compute all forces, and determine new positions using an integrator:
Verlet,
leapfrog Verlet,
velocity Verlet,
Gear predictor-corrector
…

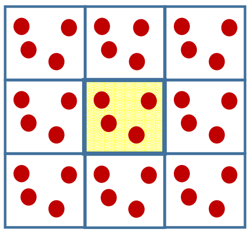
- Equilibrate: let the system reach equilibrium (i.e., lose memory of initial conditions)

- Average: accumulate quantities of interest

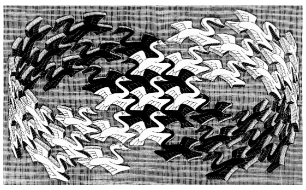
Periodic boundary conditions

- Simulations can only deal with small systems
- Surface effects are strong; but we often interested in bulk properties
- Solution: replicate system periodically

- 26 images in 3D
- Advantage: eliminates surface effects
- Disadvantage: imposes periodicity
- That can make itself felt in physical properties
- Be aware of these factors. However, free boundary conditions are typically much worse!
- Finite-size effects are investigated by simulating different system sizes (at the same density)



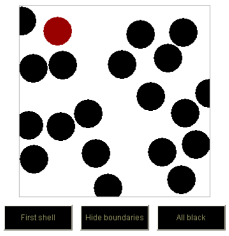
Periodic boundary conditions



Coming in from the other side

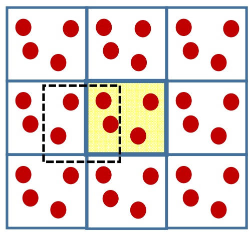
Atoms moving out from one side

Periodic boundary conditions



http://www.eng.buffalo.edu/~kofke/applets/dak_pbcCubic.html

Minimum-image convention



• If two particles are interacting, which periodic image to consider?
• Minimum image convention: only consider nearest image:
typically combined with cutoff <L/2 where L is the simulation box length.

Implantation of periodic boundary conditions

First, an atom which has passed through one face of the simulation box should re-enter the box through the opposite face:

$$\boxed{\begin{array}{ll} \text{if } (x < -L_x * 0.5) & x = x + L_x \\ \text{if } (x >= L_x * 0.5) & x = x - L_x \end{array}}$$

where L_x is the length of the box in *x* direction (assuming an orthogonal unit cell centered on the origin) and *x* is the position of the atom in the same direction.

Second, every distance between atoms should obey the minimum image criterion:

$$\boxed{\begin{array}{l} dx = x(j) - x(i) \\ dx = dx - \text{nint}(dx/L_x) * L_x \end{array}}$$

where *dx* is the distance between atom *i* and atom *j, nint(x) is* the the nearest integer function, defined as the integer closest to *x.*

For three-dimensional PBCs, both operations should be repeated in all 3 dimensions.

MD Simulation: temperature

• In a classical, many body system, the temperature is defined from the equipartition theorem in terms of the average kinetic energy per degree of freedom:

$$\frac{1}{2}m\left\langle v_x^{\,2}\right\rangle = \frac{1}{2}k_B T$$

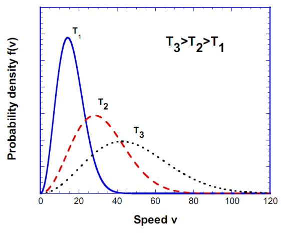
• Thus the lower the temperature, the lower the kinetic energy, and the slower the average velocity.

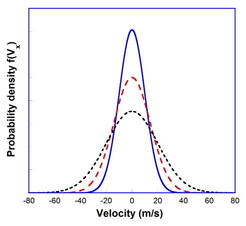
• In a simulation, we use this equation as an operational definition of T, so that for N_f degrees of freedom in the kinetic energy, the temperature function may be given as:

$$T = \frac{1}{N_f k_B}\sum_{i,\alpha} m_i v_{i,\alpha}^2$$

If we have imposed the condition of zero linear momentum in the simulations, we thus have $N_f = 3N-3$.

Maxwell-Boltzmann distribution



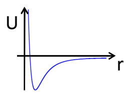


$$f(v) = 4\pi\left(\frac{m}{2\pi k_B T}\right)^{3/2} v^2 \exp\left[\frac{-mv^2}{2k_B T}\right] \qquad f(v_x) = \sqrt{\frac{m}{2\pi k_B T}}\exp\left[\frac{-mv_x^2}{2k_B T}\right]$$

MD Simulation: Finite Difference Methods

- Essential idea: the integration is broken down into many small steps, each separated in time by a fixed time δt.
- The total force on particle i at time t, $\mathbf{F}_i(t)$, is calculated as the vector sum of the individual forces $\mathbf{F}_{ij}(t)$ on i due to every particle j within its range of interaction at time t:

$$\boxed{\mathbf{F}_\mathbf{i}(t) = \sum_j \mathbf{F}_\mathbf{ij}(t)}$$



MD Simulation: Finite Difference Methods

- From the total force acting on particle i, we can find its acceleration, which -- combined with $\mathbf{r}_i(t)$ and $\mathbf{v}_i(t)$ – gives the new positions and velocities at time t+δt.

$$\mathbf{r}_i(t+\delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\delta t + \frac{1}{2}\mathbf{a}_i(t)\delta t^2$$
$$\mathbf{v}_i(t+\delta t) = \mathbf{v}_i(t) + \mathbf{a}_i(t)\delta t$$
$$\mathbf{a}_i(t) = \frac{F_i(t)}{m_i}$$

- The force is assumed to be constant during the time step δt (which means we must always choose our time step to be small enough that this is true.)

- After all the particles positions and velocities have been updated to their new values at t+δt, the force on every particle is again evaluated to determine the subsequent particle positions and velocities at t+2δt, and so on…

Integration Algorithms

- All algorithms start from assumption that the positions, velocities, and accelerations can be approximated by a Taylor series expansion:

$$\mathbf{r}(t+\delta t)=\mathbf{r}(t)+\mathbf{v}(t)\delta t+\frac{1}{2}\mathbf{a}(t)(\delta t)^2+\frac{1}{6}\mathbf{b}(t)(\delta t)^3+\frac{1}{24}\mathbf{c}(t)(\delta t)^4+\dots$$

$$\mathbf{v}(t+\delta t)=\mathbf{v}(t)+\mathbf{a}(t)\delta t+\frac{1}{2}\mathbf{b}(t)(\delta t)^2+\frac{1}{6}\mathbf{c}(t)(\delta t)^3+\dots$$

$$\mathbf{a}(t+\delta t)=\mathbf{a}(t)+\mathbf{b}(t)\delta t+\frac{1}{2}\mathbf{c}(t)(\delta t)^2+\dots$$

$$\mathbf{b}(t+\delta t)=\mathbf{b}(t)+\mathbf{c}(t)\delta t+\dots$$

The Verlet algorithm

$$\mathbf{r}_i(t+\delta t)=\mathbf{r}_i(t)+\mathbf{v}_i(t)\delta t+\frac{1}{2}\mathbf{a}_i(t)(\delta t)^2+\dots$$ Taylor expansion

$+$

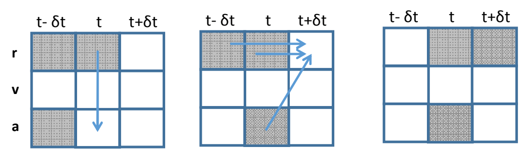
$$\mathbf{r}_i(t-\delta t)=\mathbf{r}_i(t)-\mathbf{v}_i(t)\delta t+\frac{1}{2}\mathbf{a}_i(t)(\delta t)^2-\dots$$ Time reversal

$$\Rightarrow \quad \mathbf{r}_i(t+\delta t)=2\mathbf{r}_i(t)-\mathbf{r}_i(t-\delta t)+\mathbf{a}_i(t)(\delta t)^2 \qquad \mathbf{a}_i(t)=\frac{F_i(t)}{m_i}$$

- The Verlet algorithm uses the positions and accelerations at time t, and the positions from the previous step **r**(t-δt), to calculate the new positions **r**(t+δt).

- The velocities do not explicitly appear in the Verlet integration scheme: velocities are not necessary for generating particle trajectories.

The Verlet algorithm



- To obtain the new velocities, we can calculate them from the difference between the positions at two different times:

$$\mathbf{v}_i(t)=\left[\mathbf{r}_i(t+\delta t)-\mathbf{r}_i(t-\delta t)\right]/2\delta t$$

or

$$\mathbf{v}_i(t+\frac{1}{2}\delta t)=\left[\mathbf{r}_i(t+\delta t)-\mathbf{r}_i(t)\right]/\delta t$$

The Verlet algorithm

- Advantages:

– Implementation is straightforward.
– Storage requirements are modest:
➢two sets of positions and one set of accelerations.
➢9N stored numbers, N is the number of atoms in the system.

- Disadvantages:

– Positions are calculated by adding a small term of order δt^2 to the difference of two much larger terms, which may lead to a loss in precision:

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

– Velocities always lag behind positions.
– Poor stability for large δt.

The velocity Verlet algorithm

- The velocity Verlet method gives positions, velocities, and accelerations at the same time without compromising precision:

Step 1

$$\mathbf{r}_i(t+\delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\delta t + \frac{1}{2}\mathbf{a}_i(t)(\delta t)^2$$

Step 2

$$\mathbf{v}_i(t+\delta t) = \mathbf{v}_i(t) + \frac{1}{2}\left[\mathbf{a}_i(t) + \mathbf{a}_i(t+\delta t)\right]\delta t$$

Step 3

- Implemented in three stages since calculating velocities requires accelerations at both t and t+δt.

The velocity Verlet algorithm

- Actual implementation trick:

1. Positions at t+δt are calculated using information at t.

$$\mathbf{r}_i(t+\delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\delta t + \frac{1}{2}\mathbf{a}_i(t)(\delta t)^2$$

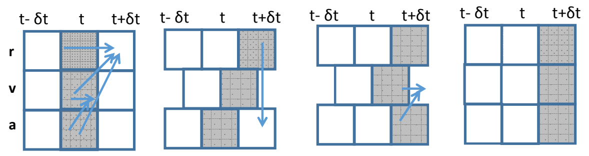
2. With second and third terms in memory, we can calculate:

$$\mathbf{v}_i(t+\frac{1}{2}\delta t) = \mathbf{v}_i(t) + \mathbf{a}_i(t)\frac{1}{2}\delta t$$

3. Calculate $\mathbf{a}_i$(t+dt) using positions at t+δt, and from this and $\mathbf{v}_i$(t+δt/2), calculate $\mathbf{v}_i$(t+δt).

$$\mathbf{v}_i(t+\delta t) = \mathbf{v}_i(t) + \frac{1}{2}\left[\mathbf{a}_i(t) + \mathbf{a}_i(t+\delta t)\right]\delta t$$

The velocity Verlet algorithm



– Same memory requirements (9N) as Verlet.

– Positions and velocities are synchronized.

– Kinetic energy (from velocities) and potential energy (from positions) can be calculated at the same time.

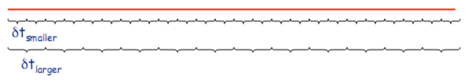
– Numerical stable, convenient and simple, the most attractive algorithm.

Which algorithm is the "best"?

Each has trade-offs:
– Fast
– Require minimal memory
– Easy to program
– reliability (Can it handle a variety of temperatures, densities, potentials?)
– Accurate: conserve energy and momentum
– Stable for large δt
…

Choosing the time step

• We want as large a δt as needed to generate a trajectory over time scales sufficient for problem.



• Larger δt, shorter computer time.

• Time step δt can't be too large or integration algorithm will be inaccurate and could be unstable.
  – If step too large, force will change too much: this causes inaccuracy.
  – If forces become too large (i.e., particles move too close together in a single time step, this causes instability.)

Choosing the time step

Practical rule-of-thumb:
THE ATOMS SHOULD NOT MOVE MORE THAN
1/20 OF THE NEAREST-NEIGHBOUR DISTANCE DURING δt

An estimate calculation of time step needed for 300 K Cu (m = 63.55 units):
From Maxwell-Boltzmann distribution, V_{rms} = 0.017 Å/fs
Nearest neighbor distance (2.55 Å) ->δt = {2.55/20}/{0.017} = 7.5 fs

In practice, for stability of the integration algorithm $\delta t < 4$ fs.

Therefore, a typical time step in MD simulation is on the order of a fs

Using modern computers it is possible to calculate 10^6 –10^8 timesteps. Therefore we can only simulate processes with MD that occur within 1 – 100 ns. This is a serious limitation for many problems that involve thermally-activated processes, cluster/vapor film deposition, annealing of irradiation damage, etc.

Timescale: limitations

In classical MD there is no way to increase the time step above ~10 fs at ordinary temperatures (77 K and up). Hence to simulate about 1 s, we would need at least 10^{14} time steps.
Realistic classical MD interatomic potentials require at least of the order of 100 computations (flops) /atom/time step.

Consider that we need to simulate a 10000 atom system:
Time step =1 fs
Processor speed = 1 Gflops/s
Calculations needed every time step =10^6 flops/time step
For a simulation of 1 μs =10^9 fs -> 10^{15} flops needed -> $10^{15}/10^9$ seconds needed -> 10^6 seconds = 11.5 days
For a simulation of 1 s =10^{15} fs -> 10^{21} flops needed -> $10^{21}/10^9$ seconds needed ->10^{12} seconds ~ **31700 days**
For a simulation of 1 s (on Blue-gene @ 1 Pflops/s) =10^{15} fs -> 10^{21} flops needed -> $10^{21}/10^{15}$ seconds needed -> 10^6 seconds ~ 11.5 days

Rensselaer at Petascale (10^{15} flops/s)

—Center for Computational Innovations

AMOS among the World's Fastest and Most Powerful Supercomputers, #97 in Nov. 2015 TOP 500 list.

http://top500.org/lists/2015/11/