Molecular Dynamics

- What to choose in an integrator
- The ***Verlet*** algorithm
- Boundary Conditions in Space and time
- Reading Assignment: **F&S Chapter 4**

The “Molecular Dynamics” (MD) method for classical systems (not H or He)

- Pick particles, masses and forces (or potential).
- Initialize positions and momentum (boundary conditions in space and time).
- Solve **F** = m **a** to determine **r**(t), **v**(t). (the integrator)
  - We need to make time discrete (t_k) not continuous
- Compute properties along the trajectory
- Estimate errors.
- Try to use the simulation to answer physical questions.

Alder & Wainwright, J Chem. Phys. ***27****, 1208 (1957). Hard sphere interaction*

Simple MD Program

- Initialize (state_now)
- Initialize verlet (state_now,state_old)
- Loop step=1,nsteps
  - Find v_and_f (R_now)
  - Verlet (state_now,state_old)
  - AverageProperties (state_now)
  - Thermostat(state_now)

Initialize (state=R,V)

- Set R=lattice (e.g. cubic lattice)
- Set V=random vector (mean=0, variance=kT/m)
- OR read state from previous run

Characteristics of simulations.

- Potentials are highly non-linear, with discontinuous higher derivatives either at the origin or at the box edge.
- Small changes in accuracy lead to totally different trajectories (the mixing or ergodic property). Long-time information is mostly statistical.
- We only need low accuracy because the potentials are not very realistic.
- **Universality saves us**: a badly integrated system is probably similar to our original system.
  - Not true in the field of non-linear dynamics e.g. in studying the solar system.
- CPU time is totally dominated by the calculation of forces.
  - We do not want to compute higher-order derivatives of the potential.
  - They might not exist!

What is a good measure of computer/algorithm efficiency?

- real time/computer time or CPU time/step (for fixed N).
- Memory limits are not too important.
- Energy conservation is important; roughly equivalent to time-reversal invariance. rule of thumb: allow $0.01k_BT$ fluctuation in the total energy.

Criteria for an Integrator

- simplicity (How long does it take to write and debug?)
- efficiency (How fast to advance a given system?)
- stability (what is the long-term energy conservation?)
- reliability (Can it handle a variety of temperatures, densities, potentials?)
- Compare statistical errors (going as $h^{-1/2}$) with time step errors (going as h^p where p=2,3….) **to pick the time step**:
- *Choose time step so that errors are balanced:*

  statistical error~ systematic errors.

The Verlet Algorithm

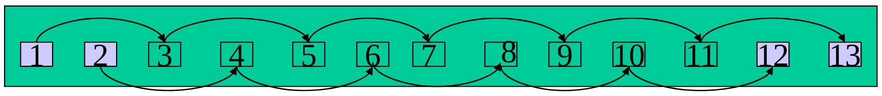
Ubiquitous choice for an integrator: Verlet (or leapfrog)

$r(t+h) = r(t) + v(t)\,h + 1/2\,a(t)\,h^2 + b(t)\,h^3 + O(h^4)$ Taylor expand

$r(t-h) = r(t) - v(t)\,h + 1/2\,a(t)\,h^2 - b(t)\,h^3 + O(h^4)$ Reverse time

$r(t+h) = 2\,r(t) - r(t-h) + a(t)\,h^2 + O(h^4)$ Add

$v(t) = (r(t+h) - r(t-h))/(2h) + O(h^2)$ *estimate* velocities



Time reversal invariance is built in: ***the energy does not drift either up or down.*** Prove this.

Historical Point of Interest: Method also suggested by **Newton** and **Newmark**!

Verlet Integrator

```
Initialize verlet (state_now,state_old)
        • Find_v_and_f (R_now)
        • r_old ←r_now-h*v_now+ ($h^2$/m)*f_now
```

```
Verlet (State_now,state_old)
    • r_new←2*r_now-r_old+($h^2$/m)*f_now
    • v_new ← (r_new-r_old)/(2*h)
    • State_old ← State_now
    • State_now ← State_new
```

How to set the time step

- Adjust to get energy conservation to 1% of kinetic energy.
- Even if errors are large, you are close to the exact trajectory of a nearby physical system with a different potential.
- Since we don't really know the potential surface that accurately, this is *often* satisfactory.
- Leapfrog algorithm has a problem with round-off error.
  - **Use the equivalent velocity-Verlet instead:**

$$\mathbf{r}(t+h) = \mathbf{r}(t) + h\,[\,\mathbf{v}(t) + (h/2)\,\mathbf{a}(t)]$$
$$\mathbf{v}(t+h/2) = \mathbf{v}(t) + (h/2)\,\mathbf{a}(t)$$
$$\mathbf{v}(t+h) = \mathbf{v}(t+h/2) + (h/2)\,\mathbf{a}(t+h)$$

Higher Order Methods?

- Higher-order does not always mean better!
- Problem is that potentials are not analytic
- Systematic error
- **Usually one tries to balance all sources of errors**

Berendsen 86

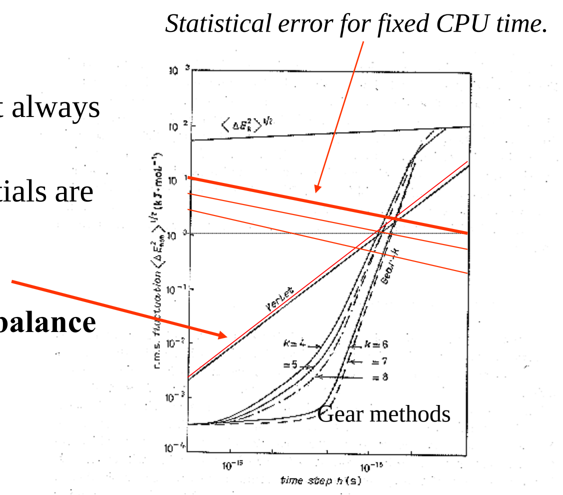


Fig. 4. – R.m.s. fluctuation of the total energy over 100 steps in a molecular-dynamics simulation of the protein BPTI (458 atoms). Verlet and Gear algorithms are compared. The fluctuation in the total kinetic energy is indicated as a reference (from ref. [15]).

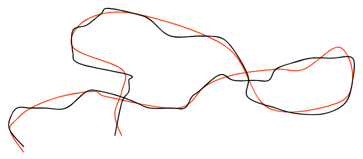
Quote from Berendsen

How accurate a simulation has to be in order to provide reliable statistical results is still a matter of debate. The purpose is never to produce reliable trajectories. Whatever accuracy the forces and the algorithms have, significant deviations from the exact solution will inevitably occur. Also, in the real physical world, individual trajectories have no meaning: in a quantum mechanical sense they do not exist and even classically they become unpredictable in the long run due to the nonisolated character of any real system. So what counts are statistical averages. It seems that very little systematic evaluation of algorithm has been done with this in mind. We have the impression that a noise as high as 10% of the kinetic energy fluctuation is still acceptable, although the accuracy of fluctuations may not be sufficient to obtain thermodynamic data from them. With such a level of inaccuracy the Verlet or leap frog algorithm is always to be preferred.

Long-term stability of Verlet

- Verlet trajectory and exact trajectory must remain close together.
- There exists another Hamiltonian, H^*, whose energy is exactly conserved by the "Verlet trajectory."

$$|H^*(P,R) - H(P,R)| \sim O(\Delta t^4)$$



• The true and pseudo energy must remain close to each other for small time-step (so-called *symplectic* behavior that conserves phase space volume).

• Hence there are no long-term drifts in the (original) energy.

• Always use sympletic algorithms- higher order ones exist.

1/31/2011 11

Spatial Boundary Conditions

Important because spatial scales are limited.

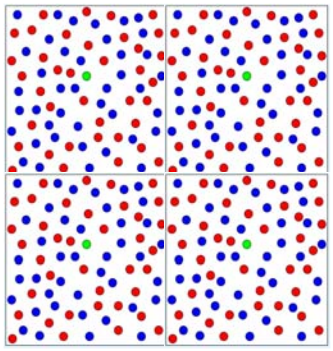
What BC can we choose? There are different possibilities:

- No boundaries; e.g. droplet, protein in vacuum.
  - If droplet has 1 million atoms and surface layer is 5 atoms thick, then 25% of atoms are on the surface.
- **Periodic Boundaries:** most popular choice because there are no surfaces (see next slide) but there can still be problems.
- Simulations on a sphere
- External potentials
- Mixed boundaries (e.g. infinite in z, periodic in x and y)

1/31/2011 12

Why use Periodic Boundary Conditions?

- Macroscopic systems $O(10^{23})$ particles.
- We eliminate *surfaces* from simulation.
- Allows us to get at *bulk properties with few particles*.
- Applied to solids, liquids, gases, and plasmas.
- Some finite-size effects remain, but can often be removed with *scaling*.

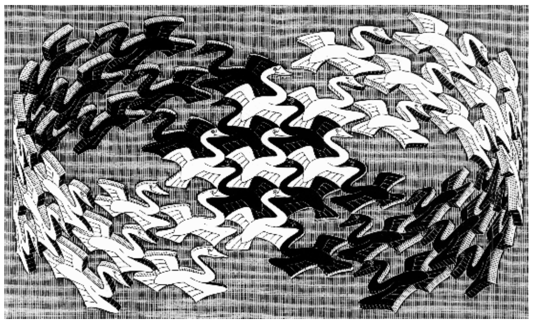


1/31/2011 13

Periodic Boundary Conditions

Is the system periodic or just an infinite array of images?

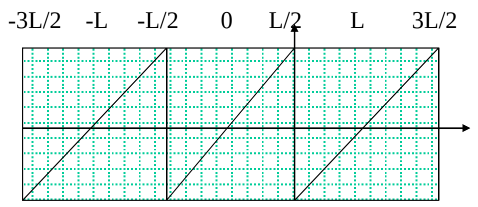
How do you calculate distances in periodic system?



1/31/2011 14

Periodic distances

Minimum Image Convention: **Take the closest distance $|r|_M = \min(r+nL)$**



How to handle the "tail" of the potential:

- *Potential cutoff* : **V(r)=0 for r > L/2**
  - continuous potential: $\mathbf{V_c(r) = V(r) - V(r_c)}$ for $r < r_c$.
  - continuous V&F: $\mathbf{V_c(r) = V(r) - V(r_c) - \nabla V(r_c)*(r-r_c)}$ for $r < r_c$.
- *Image potential*: $\mathbf{V_I = \Sigma\, v(r_i - r_j + nL)}$

For long-range potential this leads to the *Ewald image potential*.
You need a charge background and convergence factor (more later)

1/31/2011 15

LJ Force Computation

```
! Loop over all pairs of atoms.
      do i=2,natoms
        do j=1,i-1
!Compute distance between i and j.
          r2 = 0
      do k=1,ndim
             dx(k) = r(k,i) - r(k,j)
!Periodic boundary conditions.
          if(dx(k).gt. ell2(k)) dx(k) = dx(k)-ell(k)
          if(dx(k).lt.-ell2(k)) dx(k) = dx(k)+ell(k)
             r2 = r2 + dx(k)*dx(k)
          enddo
```

We can do better with the PBC!
dx(k)=dx(k)-ell(k)*nint(dx(k)/ell(k))

Why?

```
!Only compute for pairs inside radial cutoff.
          if(r2.lt.rcut2) then
            r2i=sigma2/r2
            r6i=r2i*r2i*r2i
!Shifted Lennard-Jones potential.
            pot = pot+eps4*r6i*(r6i-1)- potcut
!Radial force.
            rforce = eps24*r6i*r2i*(2*r6i-1)
            do k = 1 , ndim
              force(k,i)=force(k,i) + rforce*dx(k)
              force(k,j)=force(k,j) - rforce*dx(k)
            enddo
          endif
        enddo
      enddo
```

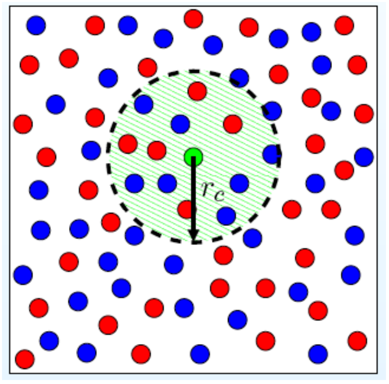
1/31/2011 16

Complexity of Force Calculations

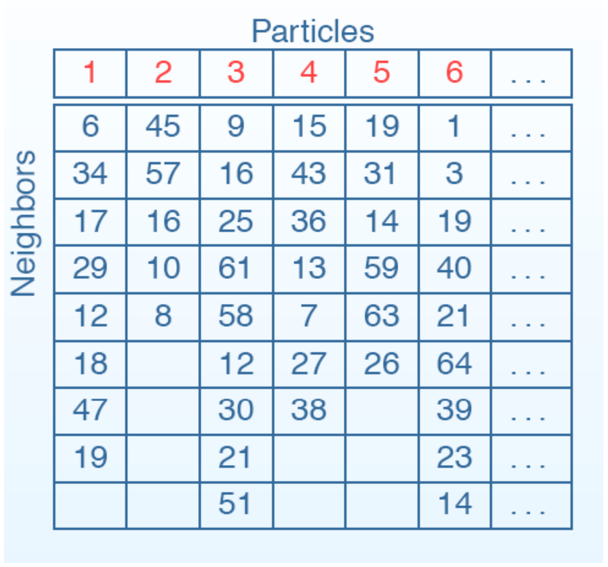
- Complexity is defined as how a computer algorithm scales with the number of degrees of freedom (atoms).
- Number of terms in pair potential is $N(N-1)/2 \approx O(N^2)$
- ***For short-range*** V(r), use *neighbor tables* to reduce it to O(N).
  - ***Explicit Neighbor list (Verlet)*** for systems that move slowly (keep a list and update occassionally.)
  - ***Bin Sort list*** (sort particles into bins and sum over neighboring bins)
- ***Long-range potentials*** require more sophisticated treatment:
  - *Ewald sums* are $O(N^{3/2})$
  - *Fast Multipole Algorithms* are O(N), for very large N.
  - Tree Methods
- More later…

Neighbor Lists: *O(N²) to O(N)*

- Pair Potentials: If $\mathbf{V_c(r) = 0}$ **for** $\mathbf{r > r_c}$ (i.e., short-ranged), then
  - **Number of computations of the potential is mN**, where ***m*** *is average number of neighbors inside* r_c and is ***independent of system size!***
  - *So, once we have the* ***neighbor table****, the calculation is O(N)!*
- Constructing Neighbor Tables:
  - **O(N²),** must check if particles fall inside each other's radius.
  - ***only have to reconstruct the table occasionally.***



A Neighbor Table



Particles

Neighbors	1	2	3	4	5	6	. . .
	6	45	9	15	19	1	. . .
	34	57	16	43	31	3	. . .
	17	16	25	36	14	19	. . .
	29	10	61	13	59	40	. . .
	12	8	58	7	63	21	. . .
	18		12	27	26	64	. . .
	47		30	38		39	. . .
	19		21			23	. . .
			51			14	. . .

1/31/2011 19

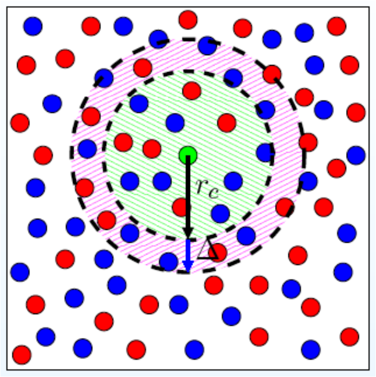
Constructing Neighbor Lists: use *skin depth*

- **Cut off table at $r_c + \Delta$** *(a skin depth)*.
  - allows a particle to move a while before new table needed.
- Keep track of the two largest distances traveled, d_1 and d_2,
- *when* $\mathbf{d_1 + d_2} \geq \Delta$ *get new table*.
- **Choose Δ to optimize efficiency**

– As Δ **decreases**, ***fewer neighbors and force evaluations***.

– As Δ **increases**, ***need to calculate neighbor table less often***. Order(N^2) operation.

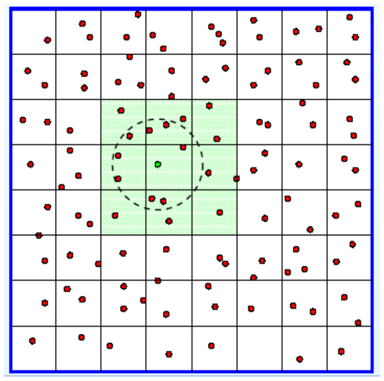
– Dynamically optimize Δ by fitting t(Δ) to a polynomial and minimizing.



1/31/2011 20

Improvements: the *Cell Method*

- **divide box in cells of size $L > r_c + \Delta$.**
  - **Need only particle's own and neighboring cells.**
- table construction is O(N)!
- To find neighbors use a linked list
- Memory is also N (not mN)

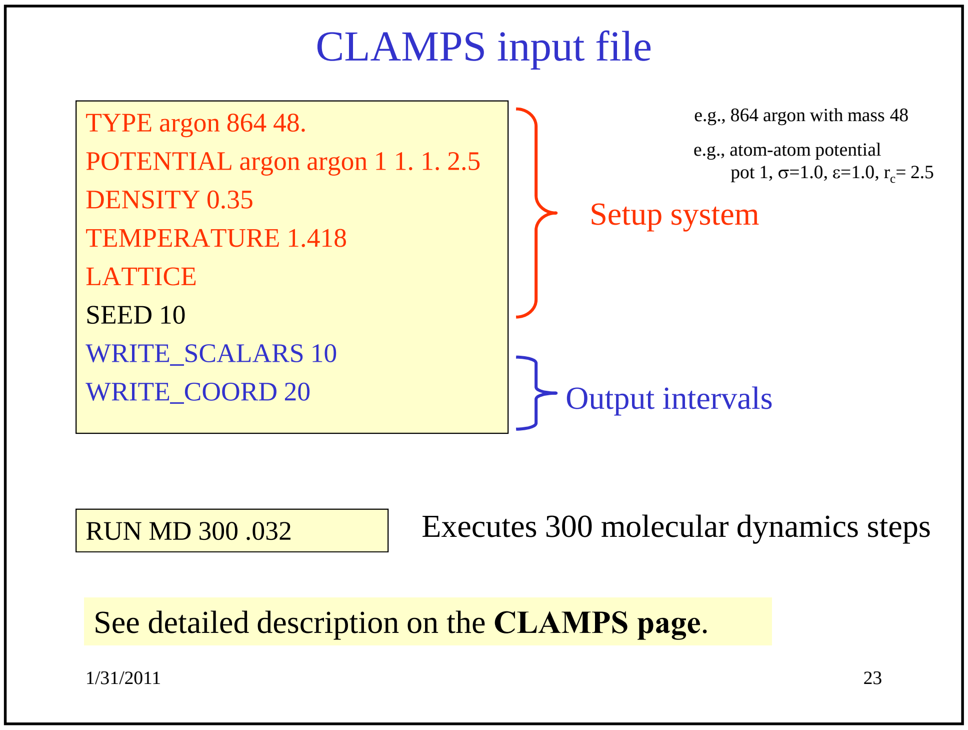


1/31/2011 21

Next time

- Potentials for MD
- HW 1 due on Monday Jan 31.

1/31/2011 22



CLAMPS input file

```
TYPE argon 864 48.
POTENTIAL argon argon 1 1. 1. 2.5
DENSITY 0.35
TEMPERATURE 1.418
LATTICE
SEED 10
WRITE_SCALARS 10
WRITE_COORD 20
```

e.g., 864 argon with mass 48

e.g., atom-atom potential
pot 1, σ=1.0, ε=1.0, r_c= 2.5

Setup system

Output intervals

```
RUN MD 300 .032
```

Executes 300 molecular dynamics steps

See detailed description on the **CLAMPS page**.

1/31/2011 23