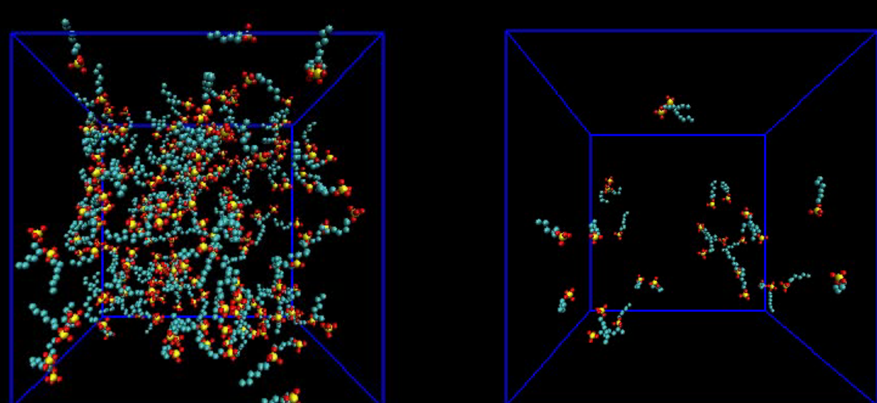
CBE 422

Molecular Dynamics Simulations

1

Atomistic S6S Micellization (H_2O not shown)



0 – 50ns   100-150ns

Sanders, Sammalkorpi & AZP, *J. Phys. Chem. B* **115**: 1403 (2011)

Molecular Dynamics (MD): main idea

- Setup system in initial configuration
- Solve Newton's equations of motion:

$$\frac{d^2\vec{r}_i}{dt^2} = \frac{\vec{F}_i}{m_i} \quad i = 1,2,\ldots,N$$

- Measure properties after system has reached equilibrium, e.g.

Energy, pressure, pair correlation functions

Similarities and differences with Monte Carlo (MC)

Both MD and MC:

- use small systems, periodic boundary conditions, sample from relevant statistical mechanical ensembles
- subject to time scale/ergodicity constraints

Differences:

- Natural ensemble for MD is microcanonical (*const.-E)*, MC easier at const. *T*
- Actual dynamic evolution of system studied in MD, phase space sampling for MC
- Direct calculation of transport coeff. in MD
- MC more flexible, unphysical moves can be used to accelerate sampling

Simple MD code: 1. Initialization

Particle positions initialized at random or on a lattice

Velocities need to be assigned:

At equilibrium, mean-square velocity along any coordinate:

$$< v_x^2 > = < v_y^2 > = < v_z^2 > = \frac{kT}{m}$$

Equilibrium velocity distribution is Maxwell-Boltzmann:

$$\mathcal{P}(v) = \frac{\exp\left(-\frac{\beta m v^2}{2}\right)}{(2\pi m k T)^{3/2}}$$

5

Initialization (cont.)

No need to start from Maxwell-Boltzmann, velocities are quickly randomized

```
subroutine init  !  initialization of MD program
sumv(1:3) = 0.   ! sum of velocities along each coordinate
sumv2 = 0.       ! sum of squares of velocities
do i=1,Npart
   do j = 1,3
      xyz(j,i) = ran(seed)*L   ! L is the box length
      v(j,i)= ran(seed)-0.5    ! random velocity
      sumv(j) = sumv(j) + v(j,i)
      sumv2 = sumv2 + v(j,i)*v(j,i)
   enddo
enddo
sumv(1:3) = sumv(1:3)/Npart
fs = sqrt(3*(Npart-1)*T/sumv2) ! because COM does not move
! set COM motion to zero and mean kinetic T to desired value
v(1:3,1:Npart) = fs*(v(1:3,1:Npart) - sumv(1:3))
return
end
```

6

Initialization (cont.)

As equilibration proceeds, temperature is going to drift away from its desired value. The mean "kinetic" temperature of the system at any time is:

$$T_{kin}(t) = \sum_{i=1}^{N} \frac{m_i u_i^2(t)}{3k(N-1)}$$

The factor $N-1$ appears because the center of mass (COM) is fixed in space.

Velocity rescaling shown on the previous slide can be applied at any time; however, it is not a proper way to achieve constant-temperature conditions and it does not conserve energy.

7

Simple MD code: 2. Moving the particles

We first need to evaluate the forces acting on each particle. In general, the force components are:

$$f_x(r) = -\frac{\partial U(r)}{\partial x} = -\frac{\partial r}{\partial x}\frac{\partial U(r)}{\partial r} = -\frac{\partial\sqrt{x^2+y^2+z^2}}{\partial x}\frac{\partial U(r)}{\partial r} = -\frac{x}{r}\frac{\partial U(r)}{\partial r}$$

For the LJ potential (with ε=σ=1), this becomes:

$$f_x(r) = \frac{48x}{r^2}\left(\frac{1}{r^{12}} - \frac{1}{2r^6}\right)$$

The force calculation code segment is executed millions and millions of times, so be very careful how you write it!

8

Force calculation code

```
!  Determine force f and energy en
f = 0. ; en = 0.         !  Array and scalar set to 0
do i=1,Npart-1
   do j = i+1,Npart
      dr(1:3) = xyz(1:3,i) - xyz(1:3,j)  ! 1=x, 2=y, 3=z
      dr(1:3) = dr(1:3) - nint(dr(1:3)*aL) ! aL is 1/L
      r2 = dr(1)*dr(1) + dr(2)*dr(2) + dr(3)*dr(3)
      r2i = 1./r2  ; r6i = r2i*r2i*r2i
      ff(1:3) = r2i*r6i*(r6i-0.5)
         do idim=1,3
            f(idim,i) = f(idim,i) + ff(idim)*dr(idim)
            f(idim,j) = f(idim,j) - ff(idim)*dr(idim)
         enddo
       en = en + r6i*(r6i-1.)
   enddo
enddo
f = 48.*f ; en = 4.*en
```

No cutoff or shift is applied to the code above (will lead to instability and loss of energy conservation)

Moving the particles (cont.)

Once the forces have been calculated, the equations of motions can now be integrated forward in time.

There are many possible numerical integration schemes. Molecular dynamics integrators must satisfy the following conditions:

- calculation of forces is expensive and can only be done once per time step
- long-term energy conservation is important
- time reversibility and conservation of volume in phase space are required ("symplectic integrators")

Some of the simplest methods are also the best.

Moving the particles (cont.): The Verlet algorithm

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2 + \frac{\Delta t^3}{3}\dddot{r} + O(\Delta t^4)$$

$$r(t-\Delta t) = r(t) - v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2 - \frac{\Delta t^3}{3}\dddot{r} + O(\Delta t^4)$$

summing these two expressions:

$$r(t+\Delta t) + r(t-\Delta t) = 2r(t) + \frac{f(t)}{m}\Delta t^2 + O(\Delta t^4)$$

subtracting :

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

Positions are more accurately known than velocities

11

Moving the particles (cont.): Velocity Verlet

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

$$v(t+\Delta t) = v(t) + \frac{f(t) + f(t+\Delta t)}{2m}\Delta t$$

This is equivalent to the original Verlet for the positions, but has better short-term energy conservation because of more accurate velocity calculations

12

Time steps and energy drift

For any time step, numerical errors that accumulate result in long-term "energy drift" – an exponential increase in energy for very large number of integration steps.

For Verlet integration, the practical limit for the maximum time step is

$$\Delta t = \frac{\sqrt{2}}{\omega}$$

Where ω is the fastest vibrational time scale in the system. For typical "soft matter" systems with bending / torsional potentials, this is approximately 1 - 2 fs, depending on whether explicit hydrogens and fixed bond lengths are used.

For the LJ system, the reduced time is:
[For Ar, the unit of time is ~2 ps]

$$t^* = \frac{t}{\sigma\sqrt{\varepsilon / m}}$$

Typically, $\Delta t^* = 0.002$ for stability [= 4 fs for Ar]

13

Liapunov Instability

$$\mathbf{r}(t) = f\left(\mathbf{r}^N(0), \mathbf{p}^N(0), t\right)$$

$$\mathbf{r}'(t) = f\left(\mathbf{r}^N(0), \mathbf{p}^N(0) + \varepsilon, t\right)$$

$$\left|\mathbf{r} - \mathbf{r}'\right| = \Delta\mathbf{r}(t)$$

For short times, $\Delta\mathbf{r}(t) \propto \varepsilon$

For long times, $\Delta\mathbf{r}(t) \propto \varepsilon \times \exp(\lambda t)$

λ: Liapunov exponent (positive)

14