Molecular Dynamics

I. The idea of molecular dynamics

Molecular dynamics (MD) is a method for computing equilibrium and kinetic properties of the systems, which obey the laws of classical physics. Any process with the characteristic timescale $\tau > \tau_q \approx 0.2$ ps can be described using classical physics. Therefore, except for the fluctuations in bond lengths and angles all other motions in molecular systems can be treated as classical. The main advantage of MD is its ability to produce "true" microscopic dynamics governed by the underlying energy landscape and interatomic forces. The implementation of MD is similar to actual experiment and involves the following stages:

1. Setting the parameters describing the conditions of MD simulations, such as temperature, number of atoms etc.

2. Initialization, which includes reading in the coordinates of atoms and generation of initial distribution of velocities

3. Computation of forces

4. Integration of Newton's equation of motion

5. Repeating steps 3 and 4 until MD simulations are done for the desired timescale

6. Computation of averages

Typical mistakes are associated with inadequate sampling of conformations, unphysical choice of initial conditions, and wrong choice of variables describing the system.

II. Initial conformation

After setting the values of parameters for MD simulations the initial structure is read in. It is important that this conformation does not include atom overlaps, unusual local conformations (i.e., unphysical values of ω dihedral angles), which may result in large forces causing the instability of the integrator. Another consideration to keep in mind is that the initial structure must be consistent with the purpose of simulations. For example, if the goal is to study the kinetics of relaxation to the native state from unfolded conformations, the initial structure must resemble random coil (unfolded) states. If the purpose is to probe equilibrium properties near the native state, then the initial structure must be native-like. In this case, the PDB structure may serve as starting conformation.

III. Initial distribution of velocities

The initial distribution of velocities is drawn from the Maxwell-Boltzmann distribution

$$P(v_{i,\alpha}) = \left(\frac{m}{2\pi k_B T}\right)^{\frac{1}{2}} e^{-\frac{mv_{i,\alpha}^2}{2k_B T}}, \quad (1)$$

where $v_{i,a}$ is the α (=x,y,z) component of the velocity of atom *i*. The distribution may be used to define the instantaneous temperature *T(t)* using the equipartition theorem

$$\left\langle \frac{mv_\alpha^2}{2} \right\rangle = \frac{1}{2} k_B T, \quad (2)$$

which relates the average kinetic energy with the temperature (<…> - ensemble average). Eq. (2) may be directly obtained from Eq. (1). Because the ensemble average corresponds to the average over all velocities of atoms, the instantaneous temperature *T(t)* is

$$k_B T(t) = \frac{1}{N_f} \sum_{i,\alpha} mv_{i,\alpha}^2, \quad (3)$$

where N_f is the number of degrees of freedom. Therefore, Eq. (3) allows us to compute the instantaneous temperature from the distribution of velocities. It is also clear that for a given realization of the velocity distribution, *T(t)* is not strictly equal to *T*. Although the velocities are generated using the distribution in Eq. (1) at the temperature *T*, the molecular system contains only finite number of atoms and the (instantaneous) temperature *T(t)* will deviate from *T*. To keep the temperature constant, one can rescale the velocities according to

$$v'_{i,\alpha} = \sqrt{\frac{T}{T(t)}} v_{i,\alpha}. \quad (4)$$

It is straightforward to show that the instantaneous temperature after rescaling *T'(t)*≡*T*. If rescaling is not carried out, the relative fluctuations of temperature in the system of *N* atoms is given by (for the system of about 1000 atoms the fluctuations in *T(t)* are ~ 3%.)

$$\frac{\Delta T(t)}{\langle T(t) \rangle} = \frac{\left(\langle T^2(t) \rangle - \langle T(t) \rangle^2\right)^{\frac{1}{2}}}{\langle T(t) \rangle} \sim N^{-\frac{1}{2}}. \quad (5)$$

IV. Force computation

It has been shown that the computation of forces scales with the number of atoms as N^2. Using Verlet and cell lists and cut-off techniques the scaling may be reduced to *N*.

Consider Algorithm 5 in Frenkel and Smit book (Box 1). In this algorithm the periodic boundary conditions and the cut-off distance $r_c < L/2$ are used (L is the size of unit cell). If the distance between atoms i and j, x_r, is larger or equal to $L/2$, then the atom j is translated to the nearest image with respect the atom i. This ensures that the atom i always "see" the atoms j in the sphere of the radius $L/2$. Then the cut-off condition is applied. In this algorithm Lennard-Jones potential is assumed (i.e., σ=1,ε_h=1)

$$V_{LJ}(r_{ij}) = 4\left(\frac{1}{r_{ij}^{12}} - \frac{1}{r_{ij}^{6}}\right). \tag{6}$$

Therefore, the force acting on the atom i is

$$f_{i,x} = -\frac{\partial V_{LJ}}{\partial x_i} = -\frac{\partial V_{LJ}}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial x_i} = 48\frac{x_i - x_j}{r_{ij}^2}\frac{1}{r_{ij}^6}\left[\frac{1}{r_{ij}^6} - 0.5\right] \tag{7}$$

The force $f_{j,x}$ has the opposite sign to $f_{i,x}$.

V. Integration of Newton equations of motion

In this section the Verlet algorithm is derived. Using Taylor expansion write the position $r(t+\Delta t)$ and $r(t-\Delta t)$ (here we omit the atom or component indexes) as

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2 + \frac{r'''(t)}{6}\Delta t^3 + o(\Delta t^4) \tag{8}$$

$$r(t-\Delta t) = r(t) - v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2 - \frac{r'''(t)}{6}\Delta t^3 + o(\Delta t^4). \tag{9}$$

Adding Eqs. (8) and (9) yields

$$r(t+\Delta t) = 2r(t) - r(t-\Delta t) + \frac{f(t)}{m}\Delta t^2 + o(\Delta t^4). \tag{10}$$

To propagate equations of motion the positions at t and $t-\Delta t$ as well as force at t must be known. The accuracy of the algorithm is $o(\Delta t^4)$. The velocities, which are needed to compute the kinetic energy and temperature, can be calculated using central difference,

$$v(t) = \frac{r(t+\Delta t) - r(t-\Delta t)}{2\Delta t} + o(\Delta t^2). \tag{11}$$

Note that the accuracy of computations of velocities is lower than that of coordinates. The implementation of Verlet algorithm is given in Box 2 (Algorithm 6 in Frenkel and

Smit book). Because we take into account only the interactions and forces between the atoms in the system, the total energy and momentum must be conserved. During MD simulations these two conditions must be explicitly checked.

VI. Requirements for MD integrator

There are several requirements that all MD algorithms must meet.

1. MD integration algorithms must be fast, although this requirement is not crucial, because the time spent on propagation of equations of motion is typically far less than the time needed to complete full evaluation of energy and forces.

2. The MD algorithm must be accurate. High accuracy would allow us to use larger integration steps and reduce the amount of expensive evaluations of energy and forces. Any MD algorithm is inherently approximate and, therefore, with time the deviation between the computed trajectory and actual, true trajectory always develops. This issue is related to so called Lyapunov instabilities. Consider two, almost identical initial conditions (1) and (2), which include coordinates and momenta. As both molecular systems evolve with time, the difference between current conformations $\Delta r(t) = \sum_i \left(r_{i,1}(t) - r_{i,2}(t)\right)^2$ would grow exponentially as $\Delta r(t) \sim e^{\lambda t}$, where $\lambda>0$ is a Lyapunov exponent and i is the atom index (*J. Comp. Phys.* **151**, 9 (1999)). The error in MD trajectory increases exponentially. However, it is believed that MD trajectories after the onset of Lyapunov instability follow closely "shadow" trajectory. This represent some other true MD trajectory, which pass through slightly perturbed initial conditions as compared to those that were actually used in simulations.

3. Because Newton equations are time reversible, so must be the MD algorithms. This implies that the algorithms should not be changed upon reversal $\Delta t \rightarrow -\Delta t$.

4. MD algorithms must be area preserving. The volume of the phase space ($\Delta\mathbf{r}$,$\Delta\mathbf{p}$) (where **r** and **p** are coordinates and momenta) must be constant during simulations. If the algorithm satisfies the condition of area preservation, the energy is conserved. It is important to note that long-term energy drift develops in any MD algorithm. Therefore, the algorithm selection should be also based on minimizing energy drift on a long time scale.

VII. Additional MD integrators

There are several other algorithms. The Euler algorithm is based on direct application of Taylor expansion, which implies using Eq. (8) with the terms up to Δt^2 order. The Euler

algorithm is not time reversible and does not conserve energy. The Runge-Kutta algorithm is not area or energy preserving and not time reversible and consequently it should not be used for MD applications. It is also possible to derive a version of Verlet algorithm, which explicitly propagates coordinates and velocities (a Leap-frog algorithm). Using Verlet algorithm the velocities at the half integration steps are

$$v(t+\frac{\Delta t}{2}) = \frac{r(t+\Delta t) - r(t)}{\Delta t} \tag{12}$$

$$v(t-\frac{\Delta t}{2}) = \frac{r(t) - r(t-\Delta t)}{\Delta t}. \tag{13}$$

From Eq. (12) we get

$$r(t+\Delta t) = r(t) + v(t+\frac{\Delta t}{2})\Delta t \tag{14}$$

Using Verlet algorithm for coordinates in Eq. (12) we write

$$v(t+\frac{\Delta t}{2}) = \frac{2r(t) - r(t-\Delta t) + \frac{f(t)}{m}\Delta t^2 - r(t)}{\Delta t} = v(t-\frac{\Delta t}{2}) + \frac{f(t)}{m}\Delta t \tag{15}$$

To propagate equations of motion using Leap-frog algorithm one need *f(t)*, *r(t)* and *v(t-Δt/2)*. The inherent problem of Leap-frog algorithm is that coordinates and velocities are not computed synchronously. As a result it is difficult to compute the total energy of the system. Leap-frog algorithm generates trajectories identical to Verlet algorithm.

The velocity form of Verlet algorithm (VFVA) computes the coordinates and velocities at the same time. According to VFVA the position *r(t+Δt)* is computed using Euler algorithm (Eq. (8)), but the velocity is obtained from forces at *t* and *t+Δt* as

$$r(t+\Delta t) = r(t) + v(t)\Delta t + \frac{f(t)}{2m}\Delta t^2 \tag{16}$$

$$v(t+\Delta t) = v(t) + \frac{f(t) + f(t+\Delta t)}{2m}\Delta t. \tag{17}$$

VFVA is equivalent to Verlet algorithm. Using Taylor expansion we get

$$r(t+2\Delta t) = r(t+\Delta t) + v(t+\Delta t)\Delta t + \frac{f(t+\Delta t)}{2m}\Delta t^2 \tag{18}$$

$$r(t) = r(t+\Delta t) - v(t)\Delta t - \frac{f(t)}{2m}\Delta t^2 \tag{19}$$

Adding Eqs. (18) and (19) we obtain

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

or taking into account Eq. (17)

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

which is the expression for Verlet algorithm. VFVA offers advantages over standard Verlet algorithm: **(1)** velocities and coordinates are propagated synchronously; **(2)** computation of velocities and coordinates at $t+\Delta t$ requires storing the positions and velocities only at t, whereas in the original form of Verlet algorithm the positions at $t-\Delta t$ must be retained as well. The peculiarity of VFVA is that coordinates must be advanced first (Eq. (16)), from which the forces at $t+\Delta t$ are computed. After this the computation of velocities takes place.

In Verlet algorithms, the long-term energy drift is small compared to most other algorithms. They are time reversible and area preserving. NAMD uses the velocity form of Verlet algorithm.

Box 1. Computation of forces

```
subroutine Force

en = 0                              # initialization of energy and
do i=1,npart                        # force array
  f(i) = 0
enddo
do i=1,npart-1                      # loop over all (i,j) pairs
   do j=i+1,npart                   #
        xr = x(i)-x(j)              # distance between particles i and j
        xr = xr –box*nint(xr/box)   # finding the nearest image of j, box=L
        r2=xr**2
        if(r2<rc2)then              # applying the cut-off for distance xr squared
                r2i=1/r2
                r6i=r2i**3
                ff=48*r2i*r6i*(r6i-0.5)   # computing force factor
                f(i) = f(i)+ff*xr         # adding force acting on the particle i
                f(j) = f(j) –ff*xr        # adding force acting on the particle j
                en = en + 4*r6i*(r6i-1)-ecut          # adding energy (ecut is LJ at rc)
        endif
   enddo
enddo
return
end subroutine Force
```

Box 2. Integration of equations of motions using Verlet algorithm

```
subroutine VerletIntegrator
sumv = 0                                # initializing the total momentum
sumv2 = 0                               # and kinetic energy
do i=1,npart
        xx=2*x(i)-xm(i)+f(i)*delta**2   # implementation of Verlet algorithm
        vi=(xx-xm(i))/(2*delta)         # getting the velocity of i
        sumv = sumv + vi                # adding up momenta
        sumv2 = sum2 + vi**2            # and energies
        xm(i) = x(i)                    # updating coordinates
        x(i) = xx                       #
enddo
temp = sum2/npart                       # computing instantaneous temperature
etot = en + sum2/2                      # and total energy
return
end subroutine VerletIntegrator
```

Notes: mass and Boltzmann constant are set to unity; delta is the integration time step