Chapter 4 Molecular Dynamics and Other Dynamics

Molecular dynamics is a method in which the motion of each individual atom or molecule is computed according to Newton's second law. It usually involves a large number of particles, from few tens to a thousand, or even several millions of particles. Macroscopic quantities are extracted from the microscopic trajectories of particles. It is a tool which we can use to understand macroscopic physics from an atomic point of view. Molecular dynamics has many applications—the thermodynamic properties of gas, liquid, and solid; phase transitions; structure and dynamics of macromolecules (e.g., polymers, DNA, proteins); hydrodynamical fluid flow; plasma and electrons, transport phenomena, etc.

1. Integration Schemes

Solving differential equations is one of the major topics in numerical analysis. We'll first review some of the methods and then introduce some of the algorithms particularly suitable for molecular dynamics.

1–1. Runge-Kutta method

Our task is to solve the differential equation

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0. \tag{1}$$

That is, given the function $f(t, y)$, find the function $y(t)$ such that it passes through the point (t_0, y_0), whose derivative is the given function. Any set of ordinary differential equations can be cast in this form if we allow y to be a vector of certain dimensions. For example, a second-order differential equation can be rewritten as a set of two first-order differential equations. Thus it is suffice to just discuss how to solve the above equation, keeping in mind that y could mean $(y_1,\, y_2,\, \cdots,\, y_d)$.

Clearly, the most obvious scheme to solve the above equation is to replace the differential by finite differences, $dt \to h$ and $dy \to y(t+h) - y(t)$. We then get the Euler method or first-order Runge-Kutta formula

$$y(t+h) = y(t) + hf\big(t, y(t)\big) + O(h^2). \tag{2}$$

The term first order refers to the fact that the equation is accurate to first order in the small step size h, thus the (local) truncation error is of order h^2. The Euler method is not recommended for practical use, because it is less accurate in comparison to other methods and it is not very stable.

The accuracy of the approximation can be improved by evaluating the function f at two points, once at the starting point, and once at the midpoint. This lead to the second-order Runge-Kutta or midpoint method.

$$\begin{aligned} k_1 &= hf\big(t, y(t)\big), \\ k_2 &= hf\big(t + \tfrac{1}{2}h,\, y(t) + \tfrac{1}{2}k_1\big), \\ y(t+h) &= y(t) + k_2 + O(h^3). \end{aligned} \tag{3}$$

However, the most popular Runge-Kutta formula is the following fourth-order one:

$$\begin{aligned} k_1 &= hf\big(t, y(t)\big), \\ k_2 &= hf\Big(t + \frac{h}{2},\, y(t) + \frac{k_1}{2}\Big), \\ k_3 &= hf\Big(t + \frac{h}{2},\, y(t) + \frac{k_2}{2}\Big), \\ k_4 &= hf\big(t + h,\, y(t) + k_3\big), \\ y(t+h) &= y(t) + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5). \end{aligned} \tag{4}$$

The fourth-order Runge-Kutta method is suitable for many applications for systems involving few degrees of freedom. But it is almost never used for molecular dynamics involving many degrees of freedom. Not only because it is computationally demanding—four evaluations of the right-hand side of the equations—it also lacks the time-reversal symmetry of the Newton's equations.

1–2. Verlet algorithm

This algorithm is particularly suited for molecular dynamics. It has been widely used in many areas from simulation of liquids and solids to biological molecules. Our problem is to solve the set of Newton's equations

$$m_i \frac{d^2\mathbf{r}_i}{dt^2} = \mathbf{F}_i, \tag{5}$$

where $\mathbf{r}_i$ is the position vector of the i-th particle, m_i is the mass of this particle, and $\mathbf{F}_i$ is the total force acting on this particle. We'll discuss the form of the forces later on.

To get approximate difference formula for derivatives, let's look at the Taylor expansion of the function $y(t)$:

$$y(t+h) = y(t) + hy'(t) + \frac{1}{2}h^2 y''(t) + \frac{1}{6}h^3 y'''(t) + O(h^4). \tag{6}$$

Replacing h by $-h$, we get

$$y(t-h) = y(t) - hy'(t) + \frac{1}{2}h^2 y''(t) - \frac{1}{6}h^3 y'''(t) + O(h^4). \tag{7}$$

If we subtract Eq. (7) from the earlier Eq. (6), we get an approximation for the first derivative,

$$y'(t) = \frac{y(t+h) - y(t-h)}{2h} + O(h^2). \tag{8}$$

When we add the equations, we found for the second derivative,

$$y''(t) = \frac{y(t+h) - 2y(t) + y(t-h)}{h^2} + O(h^2). \tag{9}$$

The same formula applies if y is a vector. Replacing the second derivative by a difference using the above approximation, Newton's equation can be written as

$$\mathbf{r}_i(t+h) = -\mathbf{r}_i(t-h) + 2\mathbf{r}_i(t) + \frac{h^2}{m_i}\mathbf{F}_i(t) + O(h^4). \tag{10}$$

This equation was used long time ago in 1791 by Delambre, and rediscovered many times, most recently by Verlet in 1960s for molecular dynamics. It is also called Stoermer algorithm. It has the property that it is exactly time-reversible. That is, if you let the system develop according to this equation for some number of steps, you can go back to your starting point *exactly* by reversing the time. This is the property of the original Newton's differential equation. If you were using the fourth-order Runge-Kutta, you do not have time-reversal symmetry exactly. This property is very important for long-time simulation.

The velocity of the particles can be computed from

$$\mathbf{v}_i(t) \approx \frac{\mathbf{r}_i(t+h) - \mathbf{r}_i(t-h)}{2h}. \tag{11}$$

One problem with Verlet algorithm is how to get started, since one needs the values at two previous times. Usually, the second-order differential equation will have a unique solution if the variables and their first derivatives are given. In our case, if the initial positions $\mathbf{r}_i(0)$ and velocities $\mathbf{v}_i(0)$ are given, our solution $\mathbf{r}(t)$ is uniquely determined. One way to start the simulation is to use

$$\mathbf{r}_i(h) = \mathbf{r}_i(0) + h\mathbf{v}_i(0) + \frac{h^2}{2m_i}\mathbf{F}_i(0). \tag{12}$$

The subsequent values $\mathbf{r}_i(2h)$, $\mathbf{r}_i(3h)$, etc, are obtained from the Verlet difference equation.

1–3. Symplectic algorithms

In the Verlet algorithm, our basic dynamic variables are the coordinates of the particles. And we have a somewhat inconvenient situation that the initial conditions are specified as the initial positions and initial velocities, while the difference scheme involves position only. As you may have learned from mechanics, it is possible to reformulate the Newtonian mechanics so that both positions and momenta are the basic dynamic variables. This formulation is called Hamiltonian dynamics (which is equivalent to Newtonian dynamics). A Hamiltonian system is completely described by the Hamiltonian function (kinetic energy plus potential energy)

$$H(p,q) = K + V, \tag{13}$$

together with Hamilton's equations of motion

$$\dot{p}_i = -\frac{\partial H}{\partial q_i}, \quad \dot{q}_i = \frac{\partial H}{\partial p_i}, \tag{14}$$

where H is considered to be a function of the generalized coordinates q_i and generalized momentum p_i. The index i labels the degree of freedom, not the particle. The symplectic algorithm can be derived if the Hamiltonian takes a simple form (see Ref. 3)

$$H = \sum_i \frac{p_i^2}{2m_i} + V(q_1, q_2, \cdots). \tag{15}$$

This is certainly the case if we use Cartesian coordinates and the interactions between particles depend only on the positions of the particles. With this particular form of Hamiltonian, Hamilton's equations can be written as

$$\dot{p}_i = -\frac{\partial V}{\partial q_i} = F_i, \quad \dot{q}_i = \frac{\partial K}{\partial p_i} = \frac{p_i}{m_i}. \tag{16}$$

The above equations reduce to Newton's equations if one eliminates the variable p_i. However, we'll use these equations to get integration schemes.

Algorithm A

We use an Euler type approximation for the momentum derivatives, and then a similar Euler algorithm for the position derivatives. However, for the positions, we use immediately the new result for the momenta. Thus, it is not exactly an Euler algorithm. For brevity, we'll use a vector notation $p = (p_1, p_2, \cdots, p_n)$ and similarly for q_i and F_i, and assume that all particles have the same mass m. With these notations and assumption, we have

$$p(t+h) = p(t) + hF\big(q(t)\big), \quad q(t+h) = q(t) + \frac{h}{m}p(t+h). \tag{17}$$

Clearly, the algorithm is accurate to first order in h.

Algorithm B

We do exactly the same thing as in algorithm A, except that we reverse the order of calculation. The new position is found first.

$$q(t+h) = q(t) + \frac{h}{m}p(t), \quad p(t+h) = p(t) + hF\big(q(t+h)\big). \tag{18}$$

If you compare algorithms A and B, you will find that it is not exactly the same. It is very interesting to note that both algorithm A and B are in fact mathematically equivalent to the Verlet algorithm if you eliminate the momenta from the equations. However, they have better roundoff error control over the plain Verlet algorithm.

Now the algorithm that we really want.

Algorithm C (second-order symplectic algorithm or velocity form of the Verlet algorithm)

This algorithm is derived by dividing the step size h into two, for the first half step size $h/2$, we use the algorithm A, for the second half of the step, we use the algorithm B. In total, we have full step size h. That is, use the formula for algorithm A but set $h/2$. We get equations relating the quantities at time t to time $t+h/2$. Then we use the equations for algorithms B with step size again $h/2$, but the time is for $t+h/2$ to $t+h$. We get four equations

$$p(t+h/2) = p(t) + \frac{h}{2}F\big(q(t)\big), \quad q(t+h/2) = q(t) + \frac{h}{2m}p(t+h/2), \tag{19}$$

$$q(t+h) = q(t+h/2) + \frac{h}{2m}p(t+h/2), \quad p(t+h) = p(t+h/2) + \frac{h}{2}F\big(q(t+h)\big). \tag{20}$$

Now we eliminate the intermediate quantities evaluated at time $t + h/2$, we get the final result for the symplectic algorithm

$$q(t+h) = q(t) + \frac{h}{m}p(t) + \frac{h^2}{2m}F\big(q(t)\big), \tag{21a}$$

$$p(t+h) = p(t) + \frac{h}{2}\Big[F\big(q(t)\big) + F\big(q(t+h)\big)\Big]. \tag{21b}$$

Note that the coordinates q have to be evaluated first since the new values at time $t+h$ are used to evaluate the forces.

The algorithm C is also second-order in step size h. From the derivation it is not obvious that it is necessarily superior than Verlet algorithm. However, the symplectic algorithm has several advantages: (1) just as the Verlet algorithm, it is time-reversible; (2) the system can be started naturally, with initial position q_0 and initial momentum $p_0 = mv_0$; (3) more importantly, the symplectic algorithms A, B, and C have one important property that share with the original Hamiltonian system—they preserve Poincaré invariants. This last property of Hamiltonian system is usually discussed in advanced mechanics or dynamical system courses. It is about the phase space (p, q) flow property. A solution with initial condition (p_0, q_0) of the Hamiltonian system $p = p(t, p_0, q_0)$, $q = q(t, p_0, q_0)$ can be viewed as a map in the phase space from the point (p_0, q_0) to the point (p, q), where the time parameter t is taken as some constant. The map is also called a canonical transformation, or symplectic transformation. One of the series of Poincaré invariants is the phase space volume. Certain region of points of volume Ω_0 can be mapped to a set of new points forming volume Ω. Invariance means $\Omega = \Omega_0$. This is called Liouville theorem. Not only is the volume in phase space invariant under the mapping of the dynamics, but also there are other lower dimensional objects (hypersurfaces) which are invariant. The symplectic algorithms preserve this invariant property exactly.

Final comment on integration algorithms: a valid algorithm should be consistent—the difference equation approaches the differential equation as the step size $h \to 0$. The algorithm should be convergent—the difference between the exact value and the approximate numerically value approaches zero as $h \to 0$. And finally it should be stable—the overall error do not grow with iterations. The last condition probably is very hard to satisfy due to the nature of (chaotic) dynamics. But it should be under reasonable control. From our experience, one would say that simple algorithm does not mean it is bad algorithm; complicated algorithm does not mean it is a good one.

2. Few-Body Problem, Planetary Motion

Before studying the more complicated many-body system of fluids, we like to test our numerical algorithms on simpler problem of few-body planetary motion. A two-body problem of the motion of the Earth and Sun is a classic example of Newtonian mechanics. As is well known, we can reduce the problem so that the Sun is fixed at the center. The other planet takes three possible types of orbits: ellipse, parabola, and hyperbola. If we add one more planet, say the Moon, then the three-body problem can no longer be solved by hand. There is no analytical solution any more.

Let's consider a numerical solution of the three-body problem consisting of the Sun which is fixed at the center of coordinate system, with mass M, the Earth with mass m and position $\mathbf{r}$, and the Moon with mass m' and position $\mathbf{r}'$. For simplicity, we assume that both the Earth and the Moon move in the same two-dimensional plane, so that the vectors will be understood as two-dimensional vectors containing just x and y components. The magnitude of the gravitational force between any pair of objects is

$$F = G\frac{mM}{r^2}, \tag{22}$$

that is, the gravitational force is proportional to the masses of the objects, and is inversely proportional to the distance square. The direction of the force is always attractive and is towards the other object. The gravitational constant is $G \approx 6.673 \times 10^{-11}\,\mathrm{m^3kg^{-1}s^{-2}}$. We have two coupled Newton's equations of motion, one for the Earth and one for the Moon:

$$m\frac{d^2\mathbf{r}}{dt^2} = -\frac{GmM\mathbf{r}}{r^3} - \frac{Gmm'(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3}, \tag{23a}$$

$$m'\frac{d^2\mathbf{r}'}{dt^2} = -\frac{Gm'M\mathbf{r}'}{r'^3} + \frac{Gmm'(\mathbf{r}-\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|^3}, \tag{23b}$$

where $r = |\mathbf{r}| = \sqrt{x^2+y^2}$. The first term of the both equations on the right-hand side is the attractive force due to the Sun, the second term is mutual interactions between the Earth and the Moon. The vector notation takes care of the magnitude as well as the directions of the forces.

Before plugging the equations into a computer, we need to change all the symbols to numbers. If we want to study the actual motions of the Sun-Earth-Moon system, we'd better use the correct constants for G, M, m, and m'. However, we should not do this naïvely, since they are very large or very small numbers that we might get overflow or underflow in calculations. It is better to normalize the basic units so that the typical quantities are not very far from 1. For this exercise, which tries to explore the general features, we can take, say, $G = 1$, $M = 100$, $m = 1$, and $m' = 0.1$. This amounts to take different units than second, meter, and kilometer and to use a particular mass ratios of the objects. With this comment in mind, it is a straightforward task to implement a numerical integration of the Newton's equations. For example, if we use the Verlet algorithm, we can follow the motion of the planets as

$$\mathbf{r}(t+h) = -\mathbf{r}(t-h) + 2\mathbf{r}(t) + h^2\left[-\frac{GM\mathbf{r}(t)}{r^3(t)} - \frac{Gm'\big(\mathbf{r}(t)-\mathbf{r}'(t)\big)}{|\mathbf{r}(t)-\mathbf{r}'(t)|^3}\right], \tag{24a}$$

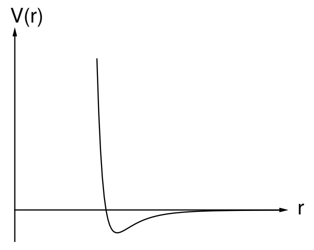
$$\mathbf{r}'(t+h) = -\mathbf{r}'(t-h) + 2\mathbf{r}'(t) + h^2\left[-\frac{GM\mathbf{r}'(t)}{r'^3(t)} + \frac{Gm\big(\mathbf{r}(t)-\mathbf{r}'(t)\big)}{|\mathbf{r}(t)-\mathbf{r}'(t)|^3}\right]. \tag{24b}$$

3. Simulation of Lennard-Jones Fluid System

3–1. Potential energy, force, and equations of motion

Lennard-Jones system is a relatively simple system to study by molecular dynamics. The principle to simulation other systems, e.g., biological molecules, is the same except that the forces between atoms become more complicated. Consider a box of point particles with equal mass m, any pair of particles has an interaction which is attractive when the distance is far, and repulsive when the particles are very close. This is described by the Lennard-Jones potential

$$V(r) = 4\,\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]. \tag{25}$$



The attractive part with a $-1/r^6$ dependence is called dispersive force (or van der Waals force) which is due to induced dipole moments of electron clouds of two atoms. The repulsive part is due to Pauli exclusion principle of electrons, which forbids two electrons to occupy the same state. The atoms behave as if they are solid balls of diameter roughly σ. The actual form of the repulsive force is not exactly $1/r^{12}$, but it is a good approximation and is also computationally more convenient.

The force acting on the second particle located at $\mathbf{r}_2$ due to the presence of the first particle located at $\mathbf{r}_1$ is

$$\mathbf{F}_{21} = -\frac{dV(r)}{dr}\hat{\mathbf{r}}, \quad \mathbf{r} = \mathbf{r}_2 - \mathbf{r}_1, \tag{26}$$

where $r = |\mathbf{r}|$ is the distance and $\hat{\mathbf{r}} = \mathbf{r}/r$ is a unit vector pointing from $\mathbf{r}_1$ to $\mathbf{r}_2$. The vector notation takes care of the direction of the force. This form of force satisfies Newton's third law—the force acting on the first particle by the second particle is $\mathbf{F}_{12} = -\mathbf{F}_{21}$. The Newton's equation for the i-th particle is

$$m\frac{d^2\mathbf{r}_i}{dt^2} = \sum_{j=1}^{N}\mathbf{F}_{ij}. \tag{27}$$

3–2. Periodic boundary conditions

In a computer simulation, we usually put the particles in a box of size L^d with periodic boundary condition. We can view the periodic boundary condition as the box repeated infinite number of times in all directions, so that each particle located at the position $\mathbf{r}_i$ have images at $\mathbf{r}_i + lL\hat{\mathbf{i}} + mL\hat{\mathbf{j}} + nL\hat{\mathbf{k}}$ (in three dimensions), where l, m, and n are integers, $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and $\hat{\mathbf{k}}$ are mutually perpendicular unit vectors of the Cartesian coordinate axes. Alternatively, we can put the system on a (three-dimensional) torus, but still use the Euclidean distance. That is, the distance between two particles is $r = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$, where Δx is the smaller one of $|x_2 - x_1|$ and $L - |x_2 - x_1|$. The shorter one when going from x_1 to x_2 (in x-direction) on a ring is used. Δy and Δz are similar defined. Care must be taken when calculating the direction of force or the vector $\mathbf{r}_2 - \mathbf{r}_1$ with periodic boundary condition.

3–3. Conservation laws

From Newtonian mechanics, we know that such a system has several conservation laws. The total momentum is conserved if there is no external force. This is the case if we simulation the system in a box with periodic boundary conditions. We should expect

$$\mathbf{P} = \sum_{i=1}^{N} m\mathbf{v}_i = \sum_{i=1}^{N} m\frac{d\mathbf{r}_i}{dt} \tag{28}$$

to be a constant. The total angular momentum,

$$\mathbf{L} = \sum_{i=1}^{N} m\mathbf{r}_i \times \mathbf{v}_i, \tag{29}$$

is also conserved if there is no external torque. Unfortunately, this quantity is not conserved if we consider the system living on a torus, why?

The most important conserved quantity in the simulation is the total energy:

$$E = K + V = \sum_{i=1}^{N} \frac{1}{2} m v_i^2 + \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} V\big(|\mathbf{r}_i - \mathbf{r}_j|\big). \tag{30}$$

This equation can be used to test your algorithm. Deviations from energy conservation will occur in simulation due to discretization and roundoff error. Note that both the Verlet algorithm and symplectic algorithms do not conserve energy exactly, even though the original Newton's differential equations respect energy conservation. However, there exists quantity which are exactly conserved, and which differs from the total energy by a higher order term in h. By the way, the total momentum conservation is obeyed exactly for the algorithms (why?).

3–4. Statistical ensembles

In the molecular dynamics simulation scheme outlined above, we have implied that the system has a fixed number of particles N in a box of volume $V = L^d$. Also the dynamics is such that the total energy E is a constant fixed at some value. In statistical mechanics, systems with fixed N, V, and E form so-called microcanonical ensemble. Our molecular dynamics simulation is thus a microcanonical one. It is possible to have canonical molecular dynamics, where the temperature T, number of particle N, and the volume are constants. We'll study those methods later on.

3–5. Micro-Macro connection

The prefix micro- refers to something that is very small, e.g., microscopic, micro-computer, and microfilm. The prefix macro- is the opposite, e.g., macroscopic, macrocosm. The ancient Greek natural philosophy is to interpret all macroscopic phenomena by the smallest constitutes of matter—atoms. This really becomes possible only in modern times. Back to our molecular dynamics. It is not very useful if you just print out the positions of thousand particles at each time step. These 'microscopic' quantities have to be somehow analyzed in some way so that we can compare them with real experiments. The theory for this micro-macro connection is statistical mechanics.

One of the most important feature of physical system (e.g., our molecular dynamical system) is that it tends to reach a state that the macroscopic quantities do not very with time if we leave the system undisturbed. This state is called an equilibrium state. Statistical mechanics studies mostly equilibrium systems. If we start our molecular system with some arbitrary initial condition, it will not be in thermodynamic equilibrium. However, if we let the system develop according to Newton's law of motion, it will get into equilibrium after sometime. Statistical mechanics has many predictions about the equilibrium properties which we'll just quote. Read your StatMech text book for more information.

(1). Microcanonical distribution (fundamental distribution)

For system at fixed N, V, and E, the probability density of finding the system in the phase space $X = (p, q) = (p_1, p_2, \cdots, q_1, q_2, \cdots)$ is a constant on the energy surface, and zero outside the energy surface. In other words, the phase space point X can be anywhere equally likely on the constant energy hypersurface. The probability being in a particular region is proportional to the volume of the region.

We can view the above statement as the basic assumption of statistical mechanics. Its validity depends on two important properties of the system: ergodicity and mixing. Complex systems like our Jennard-Jones particles in a box have these properties, while a simple system like the harmonic oscillator does not. Ergodicity means that for almost all starting point X_0 in phase space, following the dynamics of the system, it is possible to reach a neighborhood of any other point X. This allows us to replace the trajectory time average by phase space average or vice versa. This forms the basic link between statistical-mechanical theory and molecular dynamics simulation. Mixing means that the phase space points spread uniformly in the long time limit.

(2). Maxwell velocity distribution

If we look at a particular particle, its velocity will change at random. In fact, the values of velocity will obey some probability distribution. Equivalently, if we look at the velocities of all the particles at a given time, they should form the same distribution. This is the Maxwell velocity distribution

$$P(\mathbf{v}) \propto \exp\Big(-\frac{1}{2}mv^2/(k_BT)\Big), \tag{31}$$

where k_B is Boltzmann constant, and T is temperature. The proportionality constant is determined by the normalization condition

$$\int d^3\mathbf{v} P(\mathbf{v}) = 1. \tag{32}$$

We can also consider the distribution of the speed $v = |\mathbf{v}|$, which is (in three dimensions)

$$f(v)\,dv = 4\pi\left(\frac{m}{2\pi k_BT}\right)^{3/2} v^2 \exp\left(-\frac{mv^2}{2k_BT}\right)\,dv. \tag{33}$$

(3). Equipartition theorem

If you calculate the average kinetic energy of each degree of freedom using the Maxwell distribution, you get the equipartition theorem

$$\frac{1}{2}k_BT = \Big\langle\frac{1}{2}mv_{ix}^2\Big\rangle = \frac{1}{3}\Big\langle\frac{1}{2}mv^2\Big\rangle, \tag{34}$$

where v_{ix} is the x-component of the velocity of the i-th particle. This equation can be conveniently used to calculate the temperature of the system.

(4). Equation of state

The equation of state is a relation between volume, temperature, and pressure. We can calculate the pressure from

$$PV = Nk_BT + \frac{1}{3}\Big\langle\sum_{i<j}(\mathbf{r}_i - \mathbf{r}_j)\cdot\mathbf{F}_{ij}\Big\rangle, \tag{35}$$

where the summation runs over each pair of interactions exactly once [$N(N-1)/2$ terms]; each term is the scalar (dot) product of the pair interaction force and the distance vector. The above result is known as virial theorem.

(5). Pair correlation function $g(r)$

This is an important function which characterizes the system. Given a particle at some position, $g(r)d^3\mathbf{r}$ is the average number of particles in the volume element $dV = d^3\mathbf{r}$, located at a distance r away from the given particle. This function can be easily calculated from molecular dynamics, by taken a statistics of distances between particles.

If we know the pair correlation function, the average total potential energy and equation of state can be calculated.

$$\Big\langle\sum_{i<j} V\big(|\mathbf{r}_i - \mathbf{r}_j|\big)\Big\rangle = \frac{2\pi N^2}{V}\int_0^\infty V(r)\,g(r)\,r^2dr. \tag{36}$$

$$PV = Nk_BT - \frac{N^2}{6V}\int_0^\infty \frac{dV(r)}{dr}\,g(r)\,4\pi r^3dr. \tag{37}$$

3–6. Details of molecular dynamics implementation

In any numerical calculation, it is best to reduce quantities to dimensionless numbers. For the Lennard-Jones system, we have natural scales from the potential. The length can be measured in terms of the parameter σ. The scale for energy is ϵ. We measure temperature in unit of ϵ/k_B so that the combination k_BT/ϵ is a dimensionless number. We use the combination $\sqrt{m\sigma^2/48\epsilon}$ as the unit of time, which is proportional to the vibrational period of a two-body Lennard-Jones particle. In terms of the dimensionless variables, the Hamilton's equation for the Lennard-Jones system is

$$\frac{d\overline{\mathbf{p}}_i}{d\overline{t}} = \sum_{j\neq i}(\overline{\mathbf{r}}_i - \overline{\mathbf{r}}_j)\left[\overline{r}_{ij}^{-14} - \frac{1}{2}\overline{r}_{ij}^{-8}\right], \tag{38a}$$

$$\frac{d\overline{\mathbf{r}}_i}{d\overline{t}} = \overline{\mathbf{p}}_i = \overline{\mathbf{v}}_i, \tag{38b}$$

where the reduced coordinates and time are defined as

$$\overline{\mathbf{r}}_i \equiv \frac{\mathbf{r}_i}{\sigma}, \quad \overline{t} \equiv t\sqrt{\frac{48\epsilon}{m\sigma^2}}, \quad \overline{r}_{ij} \equiv |\overline{\mathbf{r}}_i - \overline{\mathbf{r}}_j|. \tag{39}$$

For each particle we need to calculate the total force acting on it by other particles. In the following particular implementation, we did not use periodic boundary condition, rather, we put the particles in side a box with repulsive hard wall. Each particle, besides interacting with every other particles, also interacts with the walls, by the form of wall potential

$$V_{wall}(x) = \frac{a}{x^9}, \tag{40}$$

where x is the perpendicular distance to the wall. Clearly, the force quickly decreases to zero away from the wall. In two dimensions, we have four terms for each of the four walls. This is purely for a better, more realistic graphic display. It is not done in usual molecular dynamics simulation which typically use periodic boundary conditions. It is not difficult to modify the program to adopt a periodic boundary condition.

After we calculated the forces due to walls for each particle, we go through each pair of particles (i, j) exactly once. There are two forces associated with each pair of particles—the force acting on particle i from particle j and other way around. They are equal in magnitude and opposite in sign. We accumulate this force for particle i (due to particle j) and for particle j (due to particle i).

To make the program very concise, we use a one-dimensional arrays for positions, momenta (same as velocity when mass is taken to be 1), and forces. Thus, `q[0]` is the x-component of the first particle; `q[1]` is the y-component of the first particle; `q[2]` is the x-component of the second particle, and similar for other quantities. The following function calculates the force `f[]` on each particle when the particles are at locations specified by `q[]`.

```
int L;

void force( double q[], double f[], int n)
{
   double xl, xr, r2, LJforce, dx, dy;
   int  i, j;

   for (i = 0; i < n; ++i) {
      xl = q[i];
      xr = L - q[i];
      f[i] = pow(xl, -9) - pow(xr, -9);
   }

   for (i = 1; i < n/2; ++i)
      for (j = 0; j < i; ++j) {
         dx = q[2*j  ] - q[2*i  ];
         dy = q[2*j+1] - q[2*i+1];
         r2 = dx*dx + dy*dy;
         LJforce = ( pow(r2,-7) - 0.5*pow(r2,-4) );
```

```
            f[2*i  ] = f[2*i  ] - LJforce*dx;
            f[2*i+1] = f[2*i+1] - LJforce*dy;
            f[2*j  ] = f[2*j  ] + LJforce*dx;
            f[2*j+1] = f[2*j+1] + LJforce*dy;
         }
   }
```

The application of the symplectic algorithm is straightforward. The function takes the current values of positions, momenta, and forces, replaces them by the values when time is elapsed by h. Note that we do not need to calculate the forces twice. We rewrite the algorithm C in the following equivalent way (assuming mass $m_i = 1$):

$$p_{n+\frac{1}{2}} = p_n + \frac{h}{2} f(q_n), \tag{41a}$$

$$q_{n+1} = q_n + h p_{n+\frac{1}{2}}, \tag{41b}$$

$$p_{n+1} = p_{n+\frac{1}{2}} + \frac{h}{2} f(q_{n+1}). \tag{41c}$$

In the very start before this function is called, one must call the `force()` function just once to initialized the force `f[]`.

```
#define nmax 200

void symplectic(double p[], double q[], double f[], int n)
{
   const double h = 0.06;
   double t[nmax];
   int i;

   for (i = 0; i < n; ++i) {
      t[i] = p[i] + 0.5 * h * f[i];
      q[i] = q[i] + h * t[i];
   }
   force(q,f,n);
   for (i = 0; i < n; ++i)
      p[i] = t[i] + 0.5 * h * f[i];
}
```

We have not discussed how to choose the step size h. Clearly, h can not be too large, otherwise, our numerical trajectory will deviate greatly from the true path. But more importantly, the algorithm becomes unstable for large value of h. As a rule of thumb, the value h should be about 1/10 or 1/20 of the smallest vibrational period of the system. For the Lennard-Jones particle system, this means that the dimensionless h should be about 0.1 or smaller.

3–7. Phase diagram of simple fluids

The Lennard-Jones model system shows all the common phases of matter—gas, liquid, and solid. The phase diagrams of the system is sketched here. They can actually computed from the method outlined above.

4. Nosé-Hoover Molecular Dynamics

In the molecular dynamics discussed in the previous sections, the system has a fixed number of particles N, fixed volume V, and fixed total energy E. Such a system is called microcanonical. These constraints are quite natural from Newtonian-dynamical point of view. In statistical mechanics, we deal most conveniently with canonical ensemble, which has a fixed number of particles N, fixed volume V, and fixed temperature T. It is also possible to simulate such systems by molecular dynamics. Such algorithms introduce addition forces to the Newton's equations. The forces are introduced purely for the seek of producing correct ensemble, they are not real forces. Thus these systems are somewhat artificial and less fundamental. There are a number of methods which realizes canonical system, one of them is to rescale velocities at each time step. Here we'll just introduce the elegant method of Nosé-Hoover (Ref. 4).

4–1. Isothermal molecular dynamics (canonical ensemble)

The new dynamical equation proposed by Nosé and Hoover in 1984 is the following,

$$\dot{q} = \frac{p}{m}, \tag{42a}$$

$$\dot{p} = f(q) - \zeta p, \tag{42b}$$

$$\dot{\zeta} = \frac{1}{dN\tau^2} \sum_i \left[\frac{p_i^2}{mk_BT} - 1 \right]. \tag{42c}$$

The notations p and q stand for the vector $(p_1, p_2, \ldots)$ and $(q_1, q_2, \ldots)$, as before. The frictional coefficient ζ is a scalar variable introduced to realize the canonical ensemble. Note that if we set $\zeta = 0$, we reduce the equations to Hamilton's equations of motion. The temperature T appears in the equation, which sets the equilibrium temperature of the system. The parameter τ is an arbitrary constant, d is dimension, and N is the number of particles.

It can be shown that the canonical distribution augmented by a Gaussian distribution in the frictional coefficient:

$$P(p, q, \zeta) \propto \exp\left[-\frac{H(p,q)}{k_BT} - \frac{1}{2}\zeta^2\tau^2 \right], \tag{43}$$

is a steady state distribution of the Nosé-Hoover dynamics. This guarantees that the time average over the trajectories of the system is equal to the average over the canonical distribution.

We can rewrite the Nosé-Hoover dynamics in a more familiar Newtonian form:

$$m\dot{\mathbf{v}}_i = \mathbf{F}_i - \zeta m \mathbf{v}_i, \quad \dot{\zeta} = \frac{1}{\tau^2}\left[\frac{K}{K_0} - 1 \right], \tag{44}$$

$$K = \sum_{i=1}^{N} \frac{1}{2} m v_i^2, \quad K_0 = \frac{1}{2} dNk_BT. \tag{45}$$

Here K is total kinetic energy, K_0 is average kinetic energy, and d is the dimension of the system. This set of equations can be solved with an algorithm similar to that of Verlet as

$$\mathbf{r}_i(t+h) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-h) + \frac{h^2}{m}\mathbf{F}_i(t) - \frac{1}{2h}\zeta(t)\big[\mathbf{r}_i(t+h) - \mathbf{r}_i(t-h)\big], \tag{46a}$$

$$\zeta(t+h) = \zeta(t) + \frac{h}{\tau^2}\left[\frac{K(t+h/2)}{K_0} - 1 \right], \quad K(t+h/2) = \frac{m}{2h^2}\sum_i \big(\mathbf{r}_i(t+h) - \mathbf{r}_i(t)\big)^2. \tag{46b}$$

The difference scheme here is also exactly time-reversible, which means that if you start from $\mathbf{r}_i(-h)$, $\mathbf{r}_i(0)$, and $\zeta(0)$, the system develops to $\mathbf{r}_i\big((n-1)h\big)$, $\mathbf{r}_i(nh)$, and $\zeta(nh)$, after n steps with positive h. Now if you run the system backwards, namely, starting from $\mathbf{r}_i(nh)$ as the previous position, $\mathbf{r}_i\big((n-1)h\big)$, as the current position and $\zeta\big((n-1)h\big)$, applying the recursion relation after n steps with h negative, you get back exactly the starting values. This cannot be true if a fourth-order Runge-Kutta method were used. This time-reversal symmetry is the key ingredient for long-time stability of the algorithm.

In the usual microcanonical molecular dynamics, total energy is a constant. This is no longer so in the canonical dynamics. Nevertheless, there exists an analogous quantity which is a constant of the motion:

$$C = K + V + K_0(\tau\zeta)^2 + 2K_0 \int^t \zeta(t')\, dt'. \tag{47}$$

This equation can be used to check the stability of integration schemes.

4–2. Isothermal-isobaric molecular dynamics

An isothermal-isobaric ensemble is a one with fixed number of particles N, temperature T, and pressure P_0. Most experimental conditions are of this type. In an isothermal-isobaric ensemble volume is a variable. The canonical distribution is replaced by

$$P(p,q,V) \propto \exp\left(-\frac{H(p,q)+P_0 V}{k_B T}\right), \tag{48}$$

where $H = K + V$ is total energy, P_0 is the constant pressure. In this ensemble, the logarithm of the (isothermal-isobaric) partition function is $-G/k_BT$, G is Gibbs free energy.

The generalization of the Newton's equations of motion is

$$\ddot{\mathbf{x}}_i = \frac{\mathbf{F}_i}{mL} - (\zeta + 2\varepsilon)\,\dot{\mathbf{x}}_i, \tag{49a}$$

$$\dot{\varepsilon} = \frac{V(P-P_0)}{k_B T \tau_V^2}, \quad \varepsilon = \frac{\dot{L}}{L}, \tag{49b}$$

$$\dot{\zeta} = \frac{1}{\tau_T^2}\left[\frac{K}{K_0} - 1\right], \quad K_0 = \frac{1}{2} dNk_BT, \tag{49c}$$

where $\mathbf{x}_i = \mathbf{r}_i/L$ is the coordinate vector normalized by the length of the box; the volume of the box is $V = L^d$; τ_V and τ_T are two parameters; P and K are the instantaneous pressure and kinetic energy, defined by

$$PV = \frac{2}{d}\sum_i \frac{1}{2} m\mathbf{v}_i^2 + \frac{1}{d}\sum_{i<j} \mathbf{r}_{ij}\cdot\mathbf{F}_{ij}; \tag{50}$$

$$K = \sum_i \frac{1}{2} m\mathbf{v}_i^2. \tag{51}$$

5. Brownian Dynamics

Brownian dynamics is a combination of molecular dynamics and Monte Carlo. Particles move according to some equations of motion, just like molecular dynamics. However, from time to time, they subject to random forces. Those forces represent the influence of the environment. Because the system has thermal contact with the surrounding, it establishes thermal equilibrium at a fixed temperature. Thus, Brownian dynamics is another way of simulating canonical ensemble. We'll introduce the idea of Brownian dynamics briefly, and will not explain in great details, since this kind of dynamics is less fundamental from the point of view of physics, and it is also used less frequently.

5–1. A one-dimensional example

Consider a particle moving in one dimension. We may imagine that the particle is moving a medium, say water, so that there will be a dissipative frictional force $-\zeta mv$. In addition, the small water molecules produce random force $R(t)$ acting on the particle. Thus the particle performs a random motion. This kind of random motion was first observed experimentally by Brown. We can write down the Newton's equation of motion (called Langevin's equation) as

$$m\frac{dv}{dt} = R(t) - \zeta mv. \tag{52}$$

The random force is assumed to be a Gaussian distribution

$$P(R) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{R^2}{2\sigma^2}}, \tag{53}$$

The values at different times are uncorrelated

$$\big\langle R(t)R(0)\big\rangle = 2\zeta m k_B T\delta(t), \tag{54}$$

where $\delta(t)$ is the Dirac-δ function. Since the differential equation involves random variable, it is called a stochastic differential equation. Because of the random force, velocity of the particle is also a random variable. It can be shown that the velocity in fact obeys Maxwell distribution (in one dimension):

$$P(v) = \frac{1}{\sqrt{2\pi m k_B T}} e^{-\frac{v^2}{2mk_B T}}, \tag{55}$$

5–2. Many particle system

This idea can be generalized to many-particle interacting systems. We add a time-dependent stochastic force to Newton's equation,

$$m\dot{\mathbf{v}}_i = \mathbf{F}_i + \mathbf{R}_i(t). \tag{56}$$

In one possible implementation of the above scheme, the random force is not really dealt with. One simply replaces the velocity with a different value drawn from a Maxwell distribution from time to time. As in the usual microcanonical molecular dynamics, the particles move according to Newton's equations of motion. After certain time interval, the normal dynamics is interrupted. Some particle is picked up and its velocity is replaced with a new value from Maxwell distribution. This process of replacing the velocity of the particles can be thought of as collisions of the particle with another imaginary (virtual) particle, which changes the velocity. This changes the kinetic energy and thus total energy, leading to a canonical distribution of the system.

References

1. D. W. Heermann, *Computer Simulation Methods in Theoretical Physics*, 2nd ed., (Springer-Verlag, 1990).
2. M. P. Allen, D. J. Tildesley, *Computer Simulation of Liquids*, (Clarendon, Oxford 1987).
3. J. M. Sanz-Serna and M. P. Calvo, *Numerical Hamiltonian Problems*, (Chapman & Hall, 1994).
4. W. G. Hoover, *Computational Statistical Mechanics*, (Elsevier, 1991).
5. J. M. Halle, *Molecular Dynamics Simulation, Elementary Methods*, (John-Wiley & Sons, 1992).
6. D. Frenkel and B. Smit, *Understanding Molecular Simulation*, 2nd Edition, (Academic Press, 2001).