II. Notes: Molecular Dynamics

Molecular Dynamics (MD) is a technique for computing the equilibrium and transport properties of a classical many-body system. In this context, the word *classical* means that nuclear motions of the constituent particles obey the laws of classical mechanics. This is an excellent approximation for a wide range of materials. Only when we consider the translational or rotational motion of light atoms or molecules (He, H_2, D_2) or vibrational motion with a frequency ν such that $h\nu > k_BT$ ($h\nu$ corresponds to the distance between energy levels) should we worry about quantum effects.

MD is in many respects very similar to real experiments. In experiments, we prepare a sample of the material, we connect it to a measuring instrument (e.g. thermometer, viscosimeter, etc), and we measure the property of interest during a certain time interval. If our measurements are subject to statistical noise, then the longer we average, the more accurate our measurements become. In MD we follow this same approach. First, we prepare a sample: we select a model system consisting of N particles and we solve Newton's equations of motion for this system until the properties of the system no longer change on average with time (this is called to *equilibrate* the system). After equilibration, we perform the actual measurement.

1. Equations of Motion

In solving the equations of motion in MD, we consider the motion of N atoms interacting through a potential V. The equation of motion in a Lagrangian equations of motion framework is:

$$\frac{d}{dt}(\partial L/\partial \dot{q}_k)-(\partial L/\partial q_k)=0 \tag{1}$$

where the Lagrangian L is a function of the generalized coordinates q and their time derivatives

$$L=L(q,\dot{q}) \tag{2}$$

and is defined in terms of the kinetic and potential energies

$$L=K-V \quad . \tag{3}$$

If we consider a system of atoms with Cartesian coordinates $\mathbf{r}_i$ and the definitions for K and V then equation 1 becomes

$$m_i \ddot{\boldsymbol{r}}_i=\boldsymbol{f}_i \tag{4}$$

where m_i is the mass of atom i and

$$\boldsymbol{f}_i=\nabla_{r_i}L=-\nabla_{r_i}V \tag{5}$$

is the force on that atom. Equation 4 coincides with Newton's 2nd law. The equations above are also valid for the center of mass motion of molecules, with $\mathbf{f}_i$ representing the total force on molecule i.

Let us define a generalized momentum p_k conjugate of generalized coordinates q_k as

$$p_k = \partial L / \partial \dot{q}_k \quad . \tag{6}$$

In the Hamiltonian form of the equations of motion, the Hamiltonian is defined as:

$$H(\boldsymbol{p}, \boldsymbol{q}) = \Sigma_k \dot{q}_k p_k - L(\boldsymbol{q}, \dot{\boldsymbol{q}}) \quad , \tag{7}$$

and the two first-order equations of motion associated with each coordinate are:

$$\dot{q}_k = \partial H / \partial p_k \tag{8}$$

and

$$\dot{p}_k = -\partial H / \partial q_k \quad . \tag{9}$$

Here we assumed that the time derivative of q_k is some function of the momenta. For a potential that is independent of velocities and time, equation 7 reduces to the sum of kinetic and potential terms and *H* is equal to the energy. For Cartesian coordinates, Hamilton's equations become

$$\dot{\boldsymbol{r}}_i = \boldsymbol{p}_i / m_i \tag{10}$$

and

$$\dot{\boldsymbol{p}}_i = -\nabla_{r_i} V = \boldsymbol{f}_i \quad . \tag{11}$$

Computing center of mass trajectories then only involves solving either a system of 3*N* second-order differential equations, equation 4, or an equivalent set of 6*N* first-order differential equations, equations 10 and 11.

1.1. First Remark

A consequence of equation 9 is that in some circumstances a particular generalized momentum p_k may be conserved. The requirement is that *L*, and hence *H*, shall be independent of the corresponding generalized coordinate q_k.

Define the center of mass coordinates of the system as a whole as

$$\boldsymbol{P} = \Sigma_i \boldsymbol{p}_i \tag{12}$$

for the total linear momentum, and for the total angular momentum

$$\boldsymbol{L}_p = \Sigma_i \boldsymbol{r}_i \times \boldsymbol{p}_i = \Sigma_i m_i \boldsymbol{r}_i \times \dot{\boldsymbol{r}}_i \tag{13}$$

where the origin is at the center of mass. For *V* depending only on particle separations and no external field, then *V*, *H*, and *L* are independent of these coordinates, thus ***P*** and ***L*** are conserved quantities for a completely *isolated* system.

In terms of symmetry, if *H* is invariant to translation in a particular direction, then the corresponding momentum component is conserved. If the system is invariant to rotation about an axis, the corresponding angular momentum component is conserved.

1. For systems in a spherical box, total angular momentum in equation 13 will be conserved, but total linear momentum in equation 12 will not.
2. For systems in a cubical box, neither total angular momentum nor total linear momentum will be conserved.
3. For periodic boundary conditions translational invariance is preserved, and hence total linear momentum, but not total angular momentum because we cannot have a spherically symmetric periodic box.

1.2. Second remark

If V and K do not depend explicitly on time, so that

$$\partial H / \partial t = 0 \quad , \tag{14}$$

then the Hamiltonian, and thus the energy, is a constant of motion. This conservation is true even with external potentials.

1.3. Third remark

The equations of motion are reversible in time. That is, by changing the signs of all velocities or momenta, we will cause the molecules to retrace their trajectories. If the equations of motions are solved *correctly*, the computer-generated trajectories will also have this property.

1.4. Fourth remark

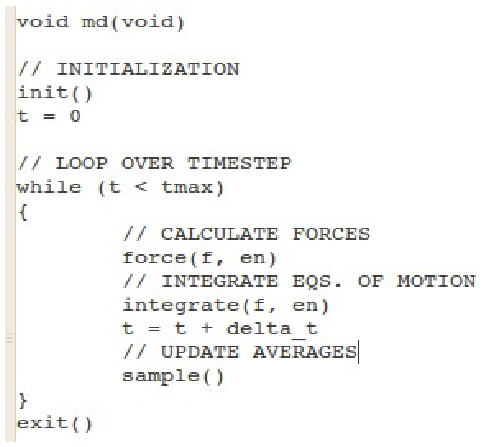
Because an explicit spatial derivative of the potential appears in the equations of motion, there is a qualitative difference in the form of the motion, and the way in which the equations are solved, depending whether V is a *continuous* function of particle positions. For particle positions that vary smoothly with time, we can use time-step methods based on Taylor's expansions of $\mathbf{r}(t)$. For a potential that varies discontinuously, as in the hard sphere and square well cases, impulsive *collisions* between particles occur at which the velocities may change discontinuously. The particle dynamics at the moment of each collision must be treated explicitly and separately from the smooth inter-collisional motion. In this case, the identification of successive collisions is the key feature for such a system.

2. An example program

Before going into the details contained in the molecular dynamics method, lets examine a very simple pseudo code of what an MD program at the very least should look like. In Figure II.1 we have a list of the shortest and simplest MD code. The program is constructed as follows:

1. Parameters that specify the conditions of the run (e.g. initial temperature, number of particles, density, time step) are first read into the program.

2. The system is initialized (i.e. the initial positions and velocities are selected) in function *init*.
3. The total forces acting on each particle are computed in function *force*.
4. The equations of motion are integrated in function *integrate.* This step and the previous one make up the core of the simulation. They are repeated until the time evolution of the system is computed for the desired length of time.
5. Thermodynamic quantities such as pressure and temperature are periodically computed in *sample.*
6. After completion of the central loop, the averages of measured quantities are computed and printed. The program stops.



```
void md(void)

// INITIALIZATION
init()
t = 0

// LOOP OVER TIMESTEP
while (t < tmax)
{
        // CALCULATE FORCES
        force(f, en)
        // INTEGRATE EQS. OF MOTION
        integrate(f, en)
        t = t + delta_t
        // UPDATE AVERAGES
        sample()
}
exit()
```

Figure II.1:

In the next section we will discuss algorithms related to the *integrate* function. Methods, theory, and algorithms for the other functions will be presented at a later point in the course.

3. Integration of Equations of Motion: Finite Difference Methods

A standard method for finding the solution to ordinary differential equations such as equations 4, 10, and 11 is the finite difference approach. The idea is the following: given molecular positions, velocities, and other dynamical properties at time *t*, we obtain as accurately as possible the positions, velocities, and other properties at a later time $t+\delta t$. The method is performed by a step by step procedure where the size of δt depends on the particular method, but in general will be significantly smaller than the typical time taken for a molecule to travel its own length.

3.1. Gear predictor-corrector method

This method follows the following general scheme:

1. predict the positions, velocities, accelerations, etc., at a time $t+\delta t$, using the current values of these quantities,
2. evaluate accelerations $\mathbf{a}_i = \mathbf{f}_i / \mathbf{m}_i$ at the new positions,

3. correct the predicted positions, velocities, accelerations, etc., using the new accelerations,
4. calculate energy, virial, order parameters, etc. before returning to the second step.

Specifically, the *predictor* step uses Taylor expansions of the form:

$$\begin{array}{lllll} \mathbf{r}^p(t+\delta t) = & \mathbf{r}(t) + & \delta t\, \mathbf{v}(t) + & \frac{1}{2}\delta t^2\mathbf{a}(t) + & \frac{1}{6}\delta t^3\mathbf{b}(t) + ... \\ \mathbf{v}^p(t+\delta t) = & \mathbf{v}(t) + & \delta t\, \mathbf{a}(t) + & \frac{1}{2}\delta t^2\mathbf{b}(t) + & ... \\ \mathbf{a}^p(t+\delta t) = & \mathbf{a}(t) + & \delta t\, \mathbf{b}(t) + & ... & \\ \mathbf{b}^p(t+\delta t) = & \mathbf{b}(t) + & ... & & \end{array} \tag{15}$$

For the "predicted" values. **b** denotes the third time derivatives of **r**. These values cannot be used as they will not generate correct trajectories as time advances because they do not utilize the equations of motion. The equations of motion enter in the *corrector* step. For the corrector step, we first calculate the true accelerations at t+δt by explicitly calculating the forces using the predicted values of position $\mathbf{r}^p$. Then we compare these calculated accelerations with the predicted ones and obtain an error expressed as

$$\Delta\mathbf{a}(t+\delta t) = \mathbf{a}^c(t+\delta t) - \mathbf{a}^p(t+\delta t). \tag{16}$$

This error is then used to calculated the "corrected" positions, etc. to obtain:

$$\begin{array}{lll} \mathbf{r}^c(t+\delta t) = & \mathbf{r}^p(t+\delta t) + & c_0\Delta\mathbf{a}(t+\delta t) \\ \mathbf{v}^c(t+\delta t) = & \mathbf{v}^p(t+\delta t) + & c_1\Delta\mathbf{a}(t+\delta t) \\ \mathbf{a}^c(t+\delta t) = & \mathbf{a}^p(t+\delta t) + & c_2\Delta\mathbf{a}(t+\delta t) \\ \mathbf{b}^c(t+\delta t) = & \mathbf{b}^p(t+\delta t) + & c_3\Delta\mathbf{a}(t+\delta t) \end{array}. \tag{17}$$

With a proper choice for the values of c_i the corrected values are now better approximations to the true positions, velocities, etc. Values of these coefficients vary depending on the number of derivatives and the order of the differential equation being solved. Note that the corrector step can be subsequently carried out with the new corrected positions iteratively for a yet better approximation. Typically, just one, and occasionally two, corrector steps are carried out for an accurate step in the MD. The least number of iterations the better as this method needs to calculate forces – the most computationally expensive operation in MD simulations.

3.2. Verlet method

The most widely used method of integrating the equations of motion is the Verlet algorithm. This method is a direct solution of the second-order differential equations 4. We first start from the Taylor expansion about **r**(t)

$$\begin{array}{lllll} \mathbf{r}(t+\delta t) = & \mathbf{r}(t) + & \delta t\, \mathbf{v}(t) + & \frac{1}{2}\delta t^2\mathbf{a}(\mathbf{t}) + & ... \\ \mathbf{r}(t-\delta t) = & \mathbf{r}(t) - & \delta t\, \mathbf{v}(t) + & \frac{1}{2}\delta t^2\mathbf{a}(\mathbf{t}) - & ... \end{array} \tag{18}$$

and add them to get

$$\mathbf{r}(t+\delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\delta t) + \delta t^2\mathbf{a}(t)\,. \tag{19}$$

Note that the velocities are not needed to compute trajectories, but they are necessary for estimating the kinetic energy (and hence total energy). They may be obtained by subtracting equations 18 to

$$\mathbf{v}(t) = \frac{\mathbf{r}(t+\delta t) - \mathbf{r}(t-\delta t)}{2\delta t} \, . \tag{20}$$

Some comments follow. First, equation 19 is correct to order δt^4 while equation 20 is correct to order δt^2. Second, this algorithm is properly centered and thus time-reversible. Third, the advancement of positions takes place all in one go rather than in two stages like in the predictor-corrector above.

In general, the Verlet algorithm has been shown to have excellent energy-conserving properties even with long time steps. Present shortcomings, such as numerical imprecision that come from the awkward handling of the velocities, are improved in the so-called 'leap-frog' scheme and later with the 'velocity Verlet' algorithm.

3.3. Velocity Verlet method

In the velocity Verlet algorithm, the original Verlet algorithm is cast in a form that uses positions and velocities computed at equal times. This velocity Verlet looks like a Taylor's expansion for the coordinates:

$$\mathbf{r}(t+\delta t) = \quad \mathbf{r}(t) + \quad \delta t \; \mathbf{v}(t) + \quad \tfrac{1}{2}\delta t^2 \mathbf{a}(\mathbf{t}) \tag{21}$$

but the update of the velocities is given by:

$$\mathbf{v}(t+\delta t) = \quad \mathbf{v}(t) + \quad \tfrac{1}{2}\delta t[\mathbf{a}(t+\delta t) + \mathbf{a}(t)] \, . \tag{22}$$

Note that here we can only compute the new velocities after we have computed the new positions and, from these, the new forces. It can be shown that this scheme is equivalent to the original Verlet algorithm.

4. Constraint Dynamics

When considering complex molecules, such as polymers, the conformational behavior of these flexible molecules is usually a complex superposition of different motions. The high frequency motions (e.g. bond vibrations) are usually of less interest than the lower frequency modes, which often correspond to major conformational changes. Unfortunately, the time step of a MD simulation is dictated by the highest frequency motion present in the system. It would therefore be of considerable benefit to be able to increase the time step without compromising the accuracy of the simulation. Constraint dynamics enable individual internal coordinates or combinations of specified coordinates to be constrained, or “fixed” during the simulation without affecting the other internal degrees of freedom. In this way the lengths of interatomic bonds can be kept constant. Such constraints have an effect on the equations of motion by the appearance of extra terms that play the role of internal forces, although these terms have an entirely different origin. The general framework comes from Lagrange's equations.

Lagrange's equations are generalized to the case of constraints by writing equation 1 as

$$\frac{d}{dt}(\partial L/\partial \dot{q}_k)-(\partial L/\partial q_k)=\Sigma_l \lambda_l a_{lk} \tag{23}$$

where λ_l are the Lagrange multipliers that need to be evaluated along with the *N* coordinates. The sum to the right of equation 23 can be regarded as a generalized force, equivalent in its effect to the imposed constraints.

For a given set of constraints that restrict a specific relationship between coordinates (holonomic)

$$g_l(\{q_k\},t)=0 \tag{24}$$

the generalized force terms have the form

$$a_{lk}=\frac{\partial g_l}{\partial q_k}\ . \tag{25}$$

Note that in a constrained system the coordinates of the particles are not independent and the equations of motion in each of the coordinate directions are connected by the constraints, such as those from equation 24.

The equations of motion for a constrained system involve two types of force: the *normal* forces arising from the intra- and intermolecular interactions (already reviewed above), and the forces due to the constraints. For bonds, we are interested in the case where the constraints g_l require the bond between atoms *i* and *j* to remain fixed. The generalized force for this constraint can be written

$$F^c_{lk}=\lambda_l \frac{\partial g_l}{\partial q_k}\quad . \tag{26}$$

Here, the constraints of interest arising from constant bond lengths are of the form

$$g_l=(\boldsymbol{r}_i-\boldsymbol{r}_j)^2-d_{ij}^2=0 \quad , \tag{27}$$

and the constraint forces lie along the direction of the bond at all times. For each constrained bond, there is an equal and opposite force on the two atoms that comprise the bond (Newton's 3rd law). The overall effect is that the constraint forces do not do work on the system. Suppose the constraint equation, labeled *l* in equation 27, corresponds to the bond length between atoms i and *j.* Then, the constraint forces are obtained by differentiating the constraint with respect to the coordinates of atoms *i* and *j* and multiplying by an undetermined multiplier

$$\partial g_l/\partial \boldsymbol{r}_i=2(\boldsymbol{r}_i-\boldsymbol{r}_j)\, so\, F^c_{li}=\lambda_l(\boldsymbol{r}_i-\boldsymbol{r}_j) \tag{28}$$

$$\partial g_l/\partial \boldsymbol{r}_j=-2(\boldsymbol{r}_i-\boldsymbol{r}_j)\, and\, F^c_{lj}=-\lambda_l(\boldsymbol{r}_i-\boldsymbol{r}_j) \quad . \tag{29}$$

The factor of 2 has been absorbed in the Lagrange multiplier. Incorporating the equations above in the Verlet expression (equation 19) for coordinate updates we obtain for a modified Verlet algorithm

$$\mathbf{r}_i(t+\delta t)=2\mathbf{r}_i(t)\ -\mathbf{r}_i(t-\delta t)+\frac{\delta t^2}{m_i}\mathbf{F}_i(t)+\Sigma_k\frac{\lambda_k\delta t^2}{m_i}\mathbf{r}_{ij}(t) \tag{30}$$

where the summation is over all constraints *k* acting on atom *i*. These constraints perturb the positions that would otherwise have been obtained from the integration algorithm, and so equation 30 can be written

$$\mathbf{r}_i(t+\delta t) = \mathbf{r}_i^o(t+\delta t) + \Sigma_k \frac{\lambda_k \delta t^2}{m_i} \mathbf{r}_{ij}(t) \qquad (31)$$

where r_i^o is the unperturbed Verlet given by equation 19.

The next step is to determine the multipliers λ_k that enable all the constraints to be satisfied simultaneously. In simple cases, a set of equations can be written for a particular atom and with the use of the bond-length constraints a quadratic equation in λ can be obtained. By dropping powers of λ higher than the linear terms, a system of equations can always be obtained and the multipliers can be determined by inverting a $k \times k$ matrix.

An alternative approach is given in the SHAKE method where each constraint is considered in turn and solved. Satisfying one constraint may cause another constraint to be violated, and so it is necessary to iterate around the constraints until they are all satisfied to within some tolerance. The tolerance should be small enough to ensure that the fluctuations in the simulation due to the SHAKE algorithm are much smaller than the fluctuations due to other sources, such as the use of cutoffs.

Other implementations of the update algorithms can be used. For example, the RATTLE method is accomplished by instead of equation 30, the velocity Verlet algorithm is used.

4.1. Comparison between Verlet and Gear

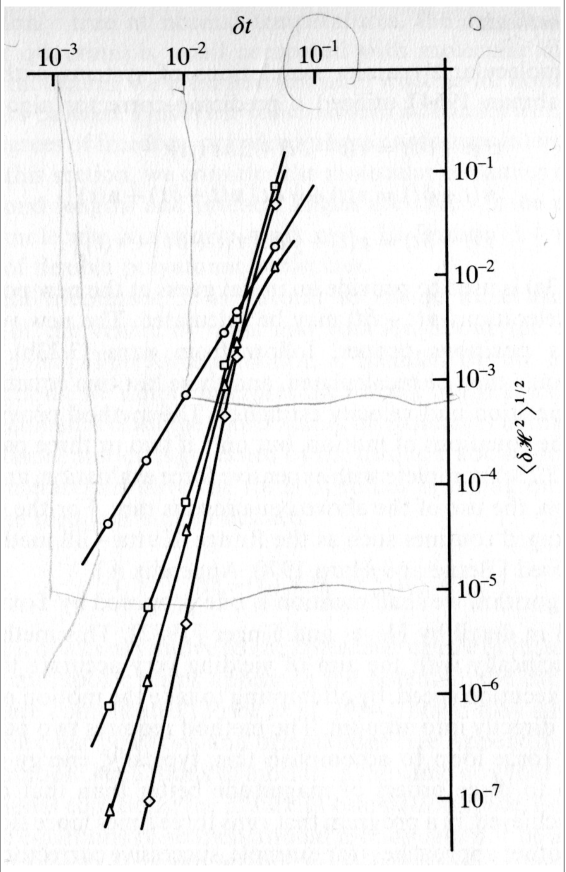


Figure II.2: Energy conservation of various algorithms. The system is a dense liquid. The y-axis is the energy fluctuations for various runs starting from the same initial conditions. The curves correspond to velocity Verlet (circles), Gear fourth-order (squares), fifth-order (triangles), and sixth-order (diamonds) algorithms.
© Oxford, from ref. 2.

One would tend to prefer the Gear predictor-corrector method for its presumed greater accuracy over the Verlet algorithm. Unfortunately, increasing the order of a Gear method does not necessarily result in a great improvement in accuracy for MD. This is because in a dense system like a liquid the forces on a molecule, and hence its motion during a time step, are dictated by the motion of its neighbors, particularly the close neighbors which move into and out of a small region of strong interaction with the molecule. This makes any Taylor series predictor, which takes no account of the motion of the neighbors, unreliable, and a high-order predictor is not a significant improvement over a low-order one. Thus, a higher order predictor-corrector method in this case has little to offer over the simpler low-order Verlet type methods.

We illustrate in Figure II.2 this point by measuring the root-mean-square energy fluctuations $\langle \delta H^2 \rangle^{1/2}$ of a dense liquid run using different δt sizes. At short δt the higher-order Gear methods are more accurate, but in most simulations we are interested in making δt as high as possible. With a longer time step, the Verlet algorithm is more attractive. However, in many applications it is more convenient to use a Gear method which can be easily adapted to handle modified first- and second- order equations of motion.

4.2. Some thoughts on important characteristics of the integration algorithm

A shortlist of these important characteristics of the integration algorithm might be as follow:

a) it should be fast,

b) it should permit the use of a long time step δt ,

c) it should duplicate the classical trajectory as closely as possible,

d) it should satisfy the known conservation laws for energy and momentum, and be time-reversible.

It turns out that for MD, not all of the points above are very important. Compared with the time-consuming force calculation, which is carried out at every time step, the raw speed of the integration algorithm is not crucial. It is more important to be able to employ a long time step δt, thus allowing for longer simulation total time. However, a longer δt will make the calculation less accurate and the solution will less likely follow the correct classical trajectory. But how important are points c) and d)?

Accuracy and Stability: First of all, it is not reasonable to expect that any approximate method of solution will dutifully follow the exact classical trajectory indefinitely. Any two classical trajectories which are initially very close will eventually diverge from one another exponentially with time.

Let the position of particle *i* be denoted by

$$\mathbf{r}_i(t) = f[\mathbf{r}(t), \mathbf{p}(t)]$$

and the perturbed position by

$$\mathbf{r}'_i(t) = f[\mathbf{r}(t), \mathbf{p}(t) + \epsilon]$$

the difference $\Delta\mathbf{r}(t)$ between both for sufficiently short times is linear in ε . However, the coefficient of the linear dependence diverges exponentially as in

$$|\Delta\mathbf{r}(t)| \sim \epsilon exp(\lambda t).$$

This so-called Lyapunov instability of the trajectories is responsible for our inability to accurately predict a trajectory for all but the shortest simulations. The exponent λ is called the Lyapunov exponent. Suppose that we wish to maintain a certain bound Δ_{max} on $\Delta\mathbf{r}(t)$, in the interval $0<t<t_{max}$. How large an initial error (ε) can we afford? From the expression above we can deduce

$$\epsilon \sim \Delta_{max} exp(-\lambda t_{max}).$$

Hence, the acceptable error in our initial conditions decreases exponentially with t_{max}, the length of the run.

In the same way, any small perturbation, even the tiny error associated with finite precision arithmetic, will tend to cause a computer-generated trajectory to diverge from the true classical trajectory with which it is initially coincident.

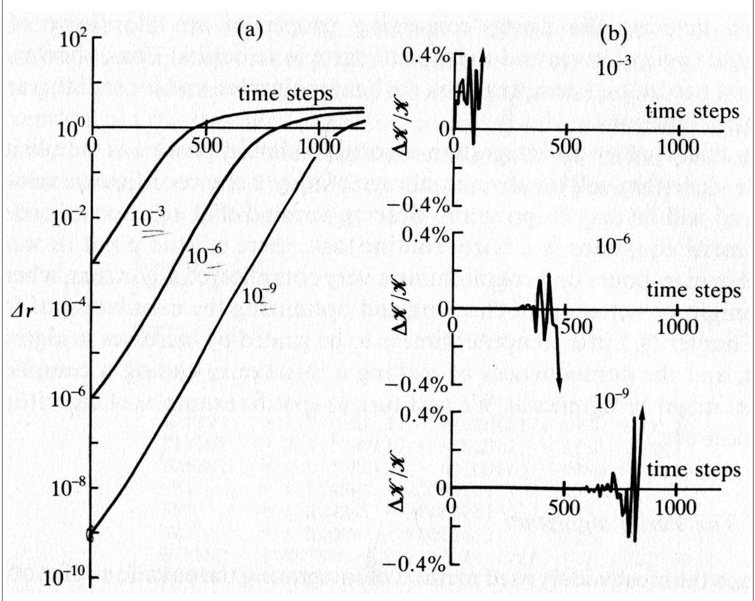


Figure II.3: Divergence of trajectories. This data come from a simulation of a dense fluid. (a) Δr *is the phase space separation between perturbed and reference trajectories. (b) ΔK/K is the percentage difference in kinetic energies. © Oxford, from ref. 2.*

In Figure II.3(a) the difference between trajectories is plotted. The quantity $|\Delta \mathbf{r}|$ where $|\Delta\mathbf{r}|^2 = (1/N)\Sigma|\mathbf{r}_i(t) - \mathbf{r}_i^o(t)|^2$ measures all the distances between particle *i* at the perturbed system, $\mathbf{r}_i$, and the unperturbed system, $\mathbf{r}_i^o$. Three perturbed systems are generated, with random displacements of value $10^{-3}\sigma$, $10^{-6}\sigma$, and $10^{-9}\sigma$ where σ is the molecular diameter. This figure shows that these small perturbations causes the trajectories to diverge from the reference trajectories in a few hundred steps. In all other respects, the simulations are identical. In (b) the different kinetic energies also become statistically uncorrelated after a short period whose length depends on the size of the initial perturbation.

These arguments illustrate that no integration algorithm will provide an essentially exact solution for a very long time, it turns out that we do not need this after all. This is because first, in MD we only need essentially exact solutions of the equations of motion for times comparable with the correlation times of interest, so that we may accurately calculate time correlation functions. Secondly, we use the method to generate states sampled from the microcanonical ensemble. We do not need exact classical trajectories to do this (hence *c* is not that important), but must lay great emphasis on energy conservation as being of primary importance for this reason (then *d* on energy is important). Momentum conservation is also important, but this can usually be easily arranged. In sum, the particle trajectories must stay on the appropriate constant-energy hypersurface in phase space, otherwise correct ensemble averages will not be generated. Energy conservation is degraded as the time step is increased, and so all simulations involve a trade-off between economy and accuracy: a good algorithm permits a large time step to be used while preserving acceptable energy conservation. Other factors dictating energy-conserving properties are the shape of the potential energy curves and the typical particle velocities. Thus, shorter time steps are used at high temperatures, for light molecules, and for rapidly varying potential functions.

5. Checks on Accuracy

The first question that should be asked after a simulation is run for the first time is: is it working properly? As a general rule, the answer is usually NO!

The first check is whether conservation laws are properly obeyed, and in particular the energy should be a *constant.* Small changes in the energy will occur, that for a simple LJ system fluctuations of order 1 part in 10^4 are generally considered to be acceptable. Energy fluctuations may be reduced by decreasing the time step. A useful suggestion is the following. Perform several short runs, each starting from the same initial configuration and covering the same total time t_{run}, but each using a different time step δt , and thus a different number of steps $\tau_{run} = t_{run}/\delta t$. The RMS energy fluctuations for each run should be calculated. If the program is functioning correctly, and other sources of energy fluctuations (such as potential truncation) have been eliminated, then the Verlet algorithm should give RMS energy fluctuations which are accurately proportional to δt^2 (see Figure II.2).

A slow upward drift of energy may also be due to a time step that is too long, to potential truncation effects, or might indicate a program error. Effects with a *physical* origin and those due to time step problems can be distinguished by running the same suggestion above. If the drift as a function of simulation time is unchanged, then it is presumably connected with the system under study, whereas if it is substantially reduced, the method used to solve the equations of motion (possibly the size of the time step) is responsible.

Catastrophic errors happening shortly after the simulation starts are evidence of program errors. Candidates for errors are: incorrect force calculation, overlapping particles, wrong units for physical constants, etc. Other errors could occur from using a potential cutoff distance that is bigger than half of the box length.

6. Computer Experiments

6.1. Thermodynamics

As stated above, MD simulations are in many respects very similar to real experiments. When we perform a real experiment, we proceed as follows. We prepare a sample of the material that we wish to study. We connect this sample to a measuring instrument (e.g. a thermometer, viscosimeter, etc), and we measure the property of interest during a certain time interval. If our measurements are subject to statistical noise (as most measurements are), then the longer we average, the more accurate our measurements becomes. In a MD simulation, we follow exactly the same approach. First, we prepare a sample: we select a model system consisting of N particles and we solve Newton's equations of motion for this system until the properties of the system no longer change with time (equilibrate). After equilibration, we perform the actual measurement.

To measure an observable quantity in a MD simulation, we must first of all be able to express this observable as a function of the positions and momenta of the particles in the system. For instance, a convenient definition of the temperature in a (classical) many-body system makes use of the equipartition of energy over all degrees of freedom that enter quadratically in the Hamiltonian of the system. In particular for the average kinetic energy per degree of freedom, we have

$$\left\langle \frac{1}{2} m v_\alpha^2 \right\rangle = \frac{1}{2} k_B T \ . \tag{32}$$

In a simulation, we use this equation as an operational definition of the temperature. In practice, we would measure the total kinetic energy of the system and divide this by the number of degrees of freedom N_f (=3N – 3 for a system of N particles with fixed total momentum). As the total kinetic energy of a system fluctuates, so does the instantaneous temperature

$$T(t) = \sum_{i=1}^{N} \frac{m_i v_i^2(t)}{k_B N_f} \ . \tag{33}$$

The relative fluctuations in the temperature will be of order $1/\sqrt{N_f}$. As N_f is typically on the order of 10^2 to 10^3, the statistical fluctuations in the temperature are on the order of 5 to 10%. To get an accurate estimate of the temperature, one should average over many fluctuations.

The pressure is defined in terms of the virial expression,

$$PV = N k_B T + \frac{1}{d} \left\langle \sum_{i=1}^{N} \mathbf{r}_i \cdot \mathbf{f}_i \right\rangle . \tag{34}$$

For pair potentials, equation 34 can be written as a sum over interacting atom pairs, namely

$$PV = N k_B T + \frac{1}{d} \left\langle \sum_{i<j} \mathbf{r}_{ij} \cdot \mathbf{f}_{ij} \right\rangle , \tag{35}$$

Where d is the dimensionality of the system, and $\mathbf{f}_{ij}$ is the force between particle i and j at a distance $\mathbf{r}_{ij}$.

6.2. Fluctuations and Response Functions

For thermodynamic quantities that are based on fluctuations, expressions vary depending on the ensemble used. Consider the constant volume specific heat, $C_v=(\partial E/\partial T)_v$. For simulations using the canonical ensemble (NVT), the specific heat is obtained by the form

$$C_v = \frac{N}{T^2}(\langle E^2 \rangle - \langle E \rangle^2). \tag{36}$$

In the microcanonical ensemble, the argument in parenthesis is equal to 0, and thus use instead the fluctuations in only the kinetic or potential energy (both are the same) for

$$C_v = \frac{3}{2}k_B \left(1 - \frac{2N(\langle E_k^2 \rangle - \langle E_k \rangle^2)}{3k_B^2 T^2}\right)^{-1}. \tag{37}$$

Note that in either ensemble we could solve numerically for $C_v = (\partial E/\partial T)_v$ by fitting the graph E(<T>).

6.3. Structural Quantities

Radial distribution and pair correlation function. The *radial distribution function*, ρ(r), is of interest for two reasons: first of all, neutron and X-ray scattering experiments on simple fluids, and light-scattering experiments on colloidal suspensions, yield information about it. Second, ρ(r) plays a central role in theories of the liquid state. Numerical results for ρ(r) can be compared with theoretical predictions and thus serve as a criterion to test a particular theory.

Consider a 2D disk or 3D spherical shell (Figure II.4) of thickness *dr* at a distance *r* from a chosen atom. The volume of the shell in 3D is approximately $4\pi r^2 dr$, where for a constant particle density system the number of atoms in the volume element varies as r^2 and thus tends to infinity as *r* tends to infinity. The *pair correlation function,* g(r), gets rid of the divergent behavior of ρ(r) by giving information about the probability of finding two particles A and B separated by distance *r*.

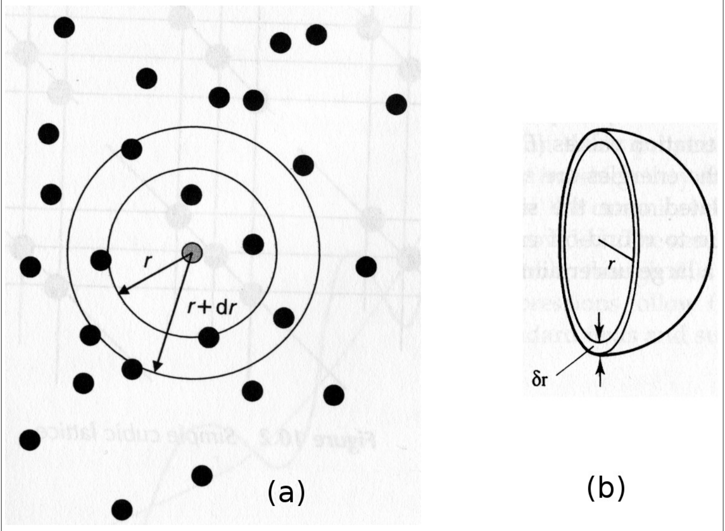


Figure II.4: Radial distribution functions use a shell to calculate number of particles as a function of radial distance. © Prentice Hall (a), © Wiley (b); from ref. 3 and 6.

In a simulation, g(r) is simply calculated as the ratio between the average number density at a distance *r* from any given atom and the density at a distance *r* from an atom in an ideal gas at the same overall density. In practice, the neighbors around each atom are sorted into distance "bins", or histograms. The number of neighbors in each bin is then averaged over the entire simulation.

In more detail, for a given time *t* we loop over each pair of particles *i* and *j* and obtain their distance r_{ij}. We then construct a histogram $n_{his}(r,t)$ that will contain the number of particle pairs within *r* and *r+dr*. Note that we can add at the same time contributions from r_{ji} which are identical. We then add contributions to this histogram from all times t_{total} to obtain $n_{his}(r)$. We normalize this histogram to obtain the average number of atoms whose distance from a given atom lies in the interval *r* and *r+dr*

$$n(r)=n_{his}(r)/(N\cdot t_{total}) \quad . \tag{38}$$

The average number of atoms in the same distance interval in an ideal gas at the same density ρ is

$$n^{ideal}(r)=4\pi\frac{\rho}{3}[(r+\delta r)^3-r^3] \quad . \tag{39}$$

By definition, the pair correlation function is then

$$g(r)=n(r)/n^{ideal}(r) \quad . \tag{40}$$

By construction, g(r) = 1 in an ideal gas. Any deviation of g(r) from unity reflects correlations between the particles due to the intermolecular interactions.

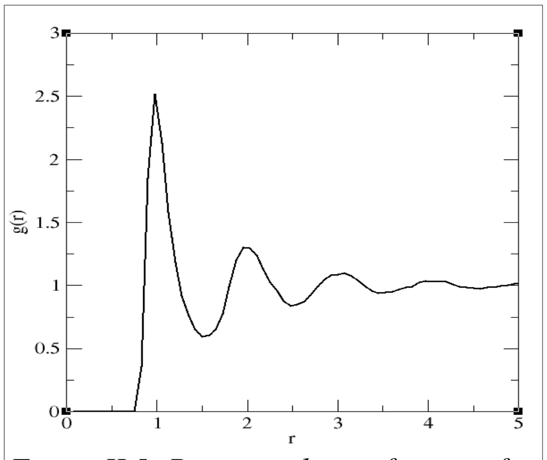


Figure II.5: Pair correlation function for a high density fluid.

In Figure II.5 we show an example pair correlation function. This distribution shows the characteristics of a dense liquid, discussed below.

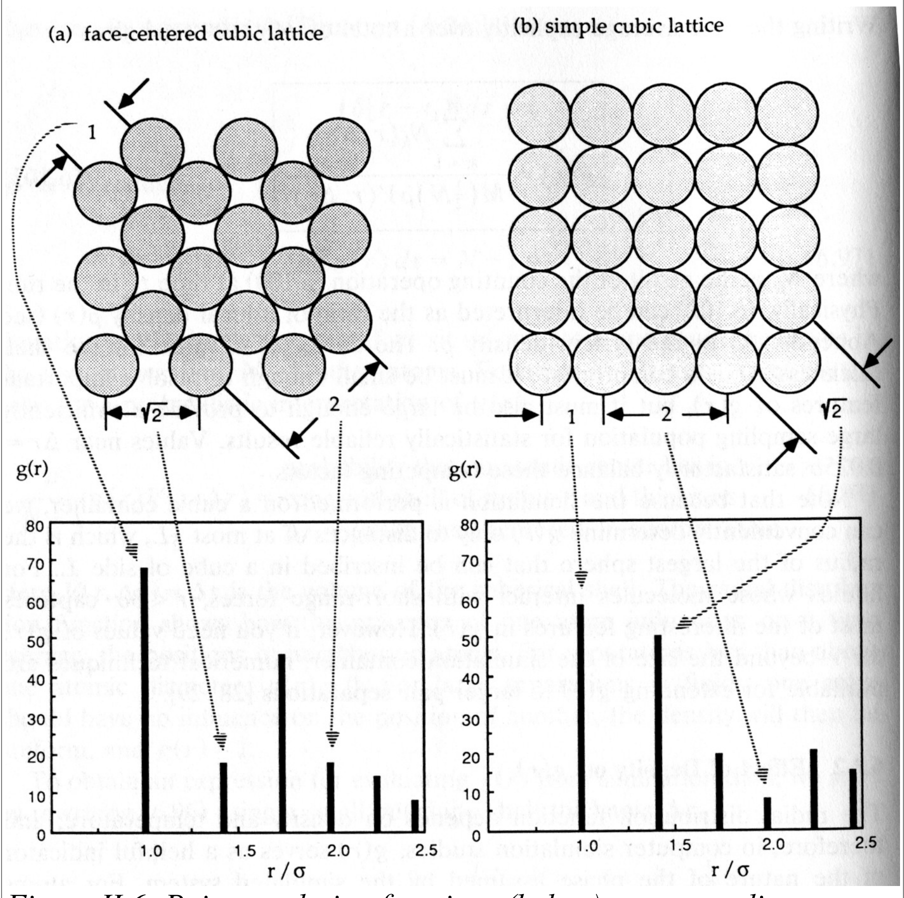


Figure II.6: Pair correlation functions (below) corresponding to different crystal lattices. © Wiley, from ref. 7.

The pair correlation function depends on density and temperature, and therefore, in computer simulation studies g(r) serves as a helpful indicator of the nature of the phase assumed by the simulated system. For atoms frozen onto the sites of regular crystal lattice structures, such as fcc, bcc, hcp, g(r) takes the form of a sequence of delta symbols. This is illustrated in Figure II.6, which compares g(r) for close-packed fcc and simple cubic lattices.

The behavior of g(r) is very different from the above when dealing with gases at low density. In a gas, atoms move freely throughout the container, interacting primarily through binary collisions, and only weak local structures form around any one atom. The g(r) function in this case can be shown to decay exponentially, as shown in Figure II.7.

For liquids and amorphous solids the behavior of g(r) is intermediate between crystal and gas: liquids exhibit *short-range* order similar to that in crystals, but *long-range* disorder like that in gases. A typical graph of g(r) for liquids is shown in Figures II.5 and II.7. The first peak can be related to the number of nearest neighbors that form a shell around any atom in the liquid, while the second peak can be related to the next nearest neighbors.

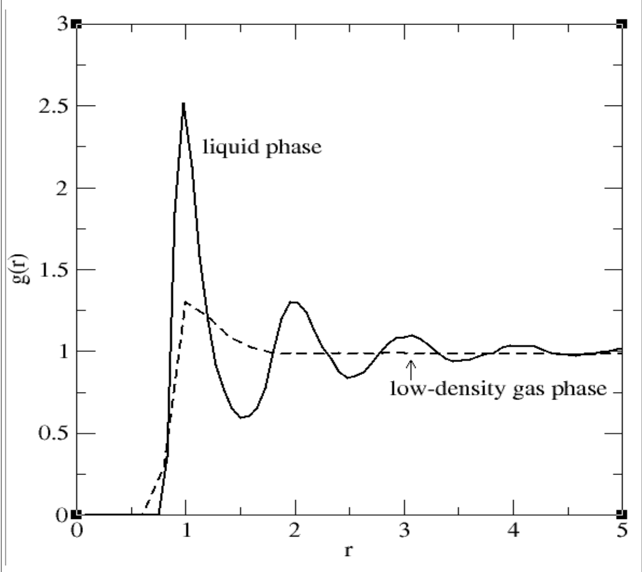


Figure II.7: Pair correlation functions for gas and liquid phases for a LJ fluid.

We can use the pair correlation function to calculate the energy and pressure of the system. The potential per particle can be calculated from

$$\begin{aligned} U/N &= \tfrac{1}{2}\rho \int_0^\infty d\mathbf{r}\ u(r)g(r) \\ &= 2\pi\rho \int_0^\infty dr\ r^2 u(r)g(r) \end{aligned} \tag{41}$$

And for the pressure from

$$\begin{aligned} P &= \rho k_B T - \tfrac{1}{3}\tfrac{1}{2}\rho^2 \int_0^\infty d\mathbf{r}\ \tfrac{du(r)}{dr} r g(r) \\ &= \rho k_B T - \tfrac{2}{3}\pi\rho^2 \int_0^\infty dr\ r^3 \tfrac{du(r)}{dr} g(r) \end{aligned} \tag{42}$$

where u(r) is the pair potential.

Equations 41 and 42 can be used to check the consistency of the energy and pressure calculations and the determinations of the pair correlation function. Care must be taken, however, since g(r) may have long tails that need to be calculated in slowly converging integrals.

6.4. Transport Coefficients

Diffusion coefficient, *D*. Diffusion of a liquid is caused by the molecular motion of the particles in the fluid. The macroscopic law that describes diffusion is know as Fick's law, which states that the flux **j** of the diffusing species is proportional to the negative gradient in the concentration of that species:

$$\mathbf{j} = -D\nabla c \ . \tag{43}$$

The diffusion of particles in a liquid with identical particles is called *self-diffusion*.

To calculate the self-diffusion coefficient, lets consider a system where the particles under study are all concentrated at the origin of the coordinate system at time t=0. To compute the time evolution of the concentration profile, we must combine equation 43 with an equation of conservation of particles,

$$\frac{\partial c(r,t)}{\partial t} + \nabla \cdot \mathbf{j}(r,t) = 0 \ . \tag{44}$$

Combining equations 43 and 44, we obtain

$$\frac{\partial c(r,t)}{\partial t} - D\nabla^2 c(r,t) = 0 \ , \tag{45}$$

that with the boundary condition c(r,0)=δ(r) we get

$$c(r,t) = \frac{1}{(4\pi Dt)^{d/2}} exp\left(-\frac{r^2}{4Dt}\right), \tag{46}$$

where d is the dimensionality of the system. By using the fact that equation 46 is normalized

$$\int d\mathbf{r}\ c(r,t) = 1 \tag{47}$$

we can express the 2nd moment (the mean-squared distance over which molecules have moved in a time interval t)

$$\langle r^2(t)\rangle \equiv \int d\mathbf{r}\ c(r,t) r^2 \ . \tag{48}$$

By multiplying equation 45 by r^2 and integrating, we get

$$\frac{\partial}{\partial t} \int d\mathbf{r}\ r^2 c(r,t) \quad = \quad D \int d\mathbf{r}\ r^2 \nabla^2 c(r,t) \tag{49}$$

where the left hand side is simply

$$\frac{\partial \left\langle r^2(t) \right\rangle}{\partial t} \ . \tag{50}$$

It can be shown that the rhs of equation 49 yields *2dD*, thus

$$\frac{\partial \left\langle r^2(t) \right\rangle}{\partial t} = 2dD \ , \tag{51}$$

obtaining a relationship between the diffusion coefficient *D* and the width of the concentration profile. This relation, that links the macroscopic to the microscopic, was first derived by Einstein. Equation 51 gives a means to calculate *D* from a simulation: for every particle *i*, we measure the distance traveled in time *t*, $\Delta\mathbf{r}_i(t)$, and plot the mean square of these distances a as function of time t given by

$$\left\langle \Delta r(t)^2 \right\rangle = \frac{1}{N} \sum_{i=1}^{N} \Delta\mathbf{r}_i(t)^2 \ . \tag{52}$$

An example of equation 52 is given in Figure II.8 (left) in which the slope of the line corresponds to *2dD*.

There is an alternative expression for the diffusion coefficient in terms of particle velocities. The expression is

$$D = \frac{1}{d} \int_0^\infty d\tau \ \langle \mathbf{v}(\tau) \cdot \mathbf{v}(0) \rangle \tag{53}$$

in which *D* is equated to the integral of the velocity autocorrelation function. An example of the velocity autocorrelation $\langle \mathbf{v}(\tau) \cdot \mathbf{v}(0) \rangle$ is shown in Figure II.8 (right).

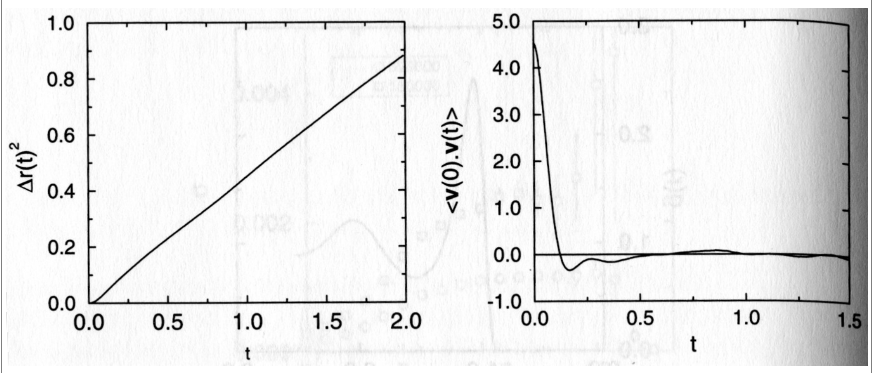


Figure II.8: (left) Mean-squared displacement as a function of the simulation time t. Note that for long times, the curve varies linearly with slope 2dD.(Right) velocity autocorrelation function as a function of simulation time. © Academic Press, ref. 1.

In general, autocorrelation functions, such as the velocity autocorrelation function, have an initial value of 1 and at long times have the value 0. The time taken to lose the correlation is often called the *correlation time*, or the *relaxation time*. If the duration of the simulation is significantly longer than the relaxation time (as it should be), then many sets of data can be extracted from the simulation to calculate the correlation function and to reduce the uncertainty in the calculation. We will discuss this point further in the next section.

The behavior of the decay of the velocity autocorrelation function depends on the density of the liquid. For a high density liquid, such as the one illustrated in Figure II.8 (right) it crosses the value 0 and into negative values. A negative value simply means that the particle is now moving in the direction opposite to that at *t=0*. This result has been interpreted in terms of a *cage* structure of the liquid; the atom hits the side of the cage formed by its nearest neighbors and rebounds, reversing the direction of its motion. At low density liquids, the velocity autocorrelation function decays monotonically to zero without structure or crossings, as there is enough space between the particles that the *cage* effect is negligible.

6.5. Error Analysis

The measurement process in MD is very similar to experiment. But the experimentalist often has the advantage of knowing that each estimate is independent, allowing well-established statistical methods to be used in the experimental data analysis. In MD, where a series of measurements is carried out in the course of a simulation of limited duration, there is no guarantee that successive estimates are sufficiently unrelated to ensure the reliability of these simple statistical methods. Averages of directly measured quantities may not be the main problem given an adequate run length, but *statistical error* estimates are particularly sensitive to correlations between samples.

There are *systematic errors* associated with, for example, finite-size effects, interaction cutoff, and the numerical integration itself; these are an intrinsic part of the computer experiment and are reproducible. There are errors due to *inadequate sampling* of phase space where, especially near a thermodynamical phase boundary, or in the case of infrequently occurring events, enough of the relevant behavior fails to be sampled; this is symptomatic of poor experimental design. And finally there is *statistical error* due to random fluctuations in the measurements; under normal circumstances this determines the degree of confidence that can be placed in the results. Only for errors of the last kind is the usual statistical analysis applicable.

In Figure II.9 we plot an example graph of the potential, kinetic, and total energy of a simulation. This figure clearly shows the equilibration time (about 1000 time steps) and the significant random fluctuations in the potential energy, the source of error last discussed in the above paragraph. When calculating static properties of the system, it is important to know the statistical error of the measurement, which is a function of the time over which the measurement was recorded. If the error is too big, then the property might not be correctly calculated, but in addition we have to know an accurate value for the error itself.

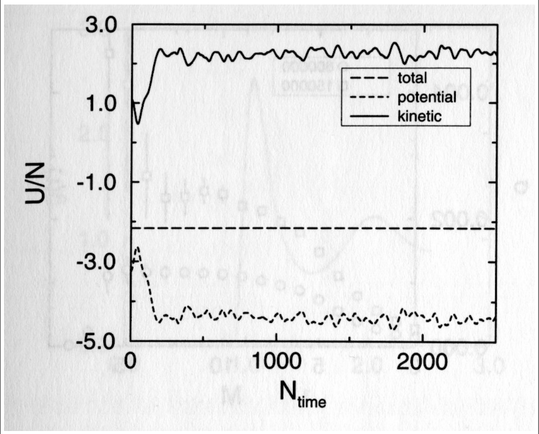


Figure II.9: Total, potential, and kinetic energy per particle as a function of the number of time steps. © Academic Press, ref. 1.

In formal terms, consider a series of *M* measurements of some fluctuating property *A*, such as the energy from Figure II.9. The mean value is

$$\langle A \rangle = \frac{1}{M} \sum_{\mu=1}^{M} A_\mu \tag{54}$$

and if each measurement A_μ is independent, with variance

$$\sigma^2(A) = \frac{1}{M} \sum_{\mu} (A_\mu - \langle A \rangle)^2 = \langle A^2 \rangle - \langle A \rangle^2 \tag{55}$$

then the variance of the mean $\langle A \rangle$ is

$$\sigma^2(\langle A \rangle) = \frac{1}{M} \sigma^2(A)\,. \tag{56}$$

But if, as is usually the case in MD simulations, the assumed independence of the A_μ is unwarranted,

$\sigma^2(\langle A \rangle)$ is liable to be underestimated because the effective number of independent measurements is considerably less than *M*.

A method to calculate the statistical error in these quantities is the *block averages* method that estimates the standard deviation. In this method, we first start calculating the standard deviation of all of the points after equilibration. Then we combine every two adjacent data points into one (average) and use this average as the new data point to calculate a new standard deviation. We will then have half the original number of points for this calculation. The procedure is repeated until there are not enough data points to calculate a standard deviation. The number of averaged points at each step is labeled M_b where each block has a time t_b, as illustrated in Figure II.10.

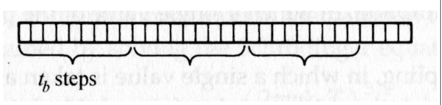


Figure II.10: Blocking a simulation to calculate the statistical error. © Academic Press, ref. 1.

This method tries to overcome the problem of correlation between data points. In our time series of data, over a short time (less than the correlation time) measurements are always correlated. If we use these data points, we will always obtain a too small standard deviation. By block averaging, there will be a number M_b below which the averaged points are no longer correlated because the size of the block is larger than the correlation time. After this value is reached, the value of the standard deviation will not change as a function of M_b. This plateau will indicate an estimated value for the standard deviation that we want.

Thus, for block sizes $b=1,2,4,\ldots$, with the upper bound being set by the total size of the data set, for each *b* the estimator for the variance can be shown to be

$$\sigma^2(\langle A \rangle_b) = \frac{1}{M_b - 1} \sum_{\beta=1}^{M_b} (A_\beta - \langle A \rangle)^2 \tag{57}$$

where M_b is the total number of blocks for that particular *b*, A_β each bock average, and $\langle A \rangle$ the overall average.

An example of this calculation is shown in Figure II.11. For small values of M (large M_b) the correlations underestimate the error until a plateau is reached. After that, the number of points (M_b small) are not enough and the error increases again to a large number. An extra bonus of this method arises from the fact that if there is not a plateau, we know that the simulation is not long enough. In addition, this graph also shows how by increasing the number of time steps by a factor of 4, the error is correspondingly diminished by a factor of 2.

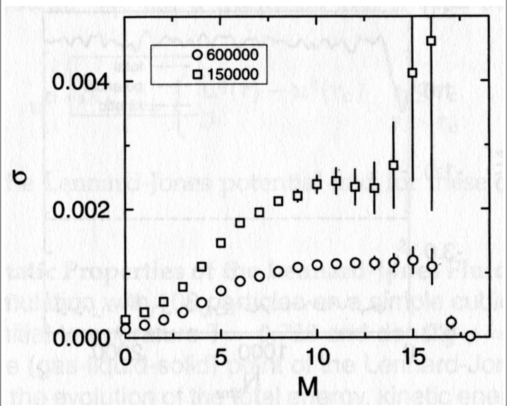


Figure II.11: Standard error in the potential energy as a function of size of block M for the number of time steps indicated by the symbols. Note that M here is not the same as M_b in the text. © Academic Press, ref. 1.

7. *Other Dynamical Systems*

7.1. Hard Spheres

Systems where particles interact via "hard" potentials (i.e. discontinuous functions of distance) must be solved in a way which is qualitatively different from the MD of soft bodies. Whenever the distance between two particles becomes equal to a point of discontinuity in the potential, a *collision* occurs in which the particle velocities will suddenly change. Thus, the primary aim of a simulation program here is to locate the time, collision partners, and all impact parameters for every collision occurring in the system in chronological order. Instead of a regular step-by-step approach, as for soft potentials, discontinuous potential programs evolve on a collision-by-collision base, computing the collision dynamics and then searching for the next collision. The general scheme is the following:

(a) locate next collision,

(b) move all particles forward until collision occurs,

(c) implement collision dynamics for the colliding pair,

(d) calculate any properties of interests, ready for averaging before returning to the first step.

In between collisions, simulations evolve in a force-free motion. For spherical molecules with hard or square-well potential interactions, the location of the time of collision between any two particles requires the solution of a quadratic equation.

In general, a program to solve hard-sphere MD has two functions to perform: the calculation of collision times and the implementation of collision dynamics. The collision time calculation is the expensive part of the program since, in principle, all possible collisions between distinct pairs must be considered.

Consider two spheres, *i* and *j*, of diameter σ, whose positions at time *t* are $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$, and whose velocities are $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$. If these particles are to collide at time $t + t_{ij}$ then the following equation will be satisfied

$$|\mathbf{r}_{ij}(t+t_{ij})| = |\mathbf{r}_{ij} + \mathbf{v}_{ij}t_{ij}| = \sigma \tag{58}$$

Where $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$ and $\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j$. If we define $b_{ij} = \boldsymbol{r}_{ij} \cdot \boldsymbol{v}_{ij}$, then this equation becomes

$$v_{ij}^2 t_{ij}^2 + 2b_{ij}t_{ij} + r_{ij}^2 - \sigma^2 = 0 \ . \tag{59}$$

If $b_{ij} > 0$, then the molecules are going away from each other and they will not collide. If $b_{ij} < 0$, it may still be true that $b_{ij}^2 - v_{ij}^2(r_{ij}^2 - \sigma^2) < 0$, in which case equation 59 has complex roots and again no collision occurs. Otherwise two positive roots arise, the smaller of which corresponds to impact

$$t_{ij} = \frac{-b_{ij} - (b_{ij}^2 - v_{ij}^2(r_{ij}^2 - \sigma^2))^{1/2}}{v_{ij}^2} \ . \tag{60}$$

We have assumed that the system knows when the next event is due. This implies the existence of a time-ordered event table. Such a table must not only produce the next event but must also be easily modifiable: the table will include many future events and, as collisions occur, changes must be made to its contents, both to incorporate newly predicted collisions and to remove previously predicted collisions that are not longer relevant because a participant has in the meantime undergone a different collision. Once the effort has been made to find a possible future collision, this information should be retained for as long as it is potentially useful, but it must be recognized that if the table includes a few collision events involving each atom, it is likely that most of this information will become obsolete before it has a chance to be used. Thus the table organization is central to the viability of this method.

7.2. Hard-sphere Polymer Chains

Flexible polymer chains – reminiscent of a bead necklace – can be constructed by placing each pair of chain neighbors in a potential well with infinitely high walls and a width corresponding to the maximum bond elongation. While this kind of model lacks the refinement of more realistic models, it allows polymer studies to benefit from the advantages of event-driven MD.

7.3. Langevin's Dynamics

Langevin's dynamics is another continuum-dynamical method based on the Langevin equations where the forces are no longer computed explicitly but are replaced by stochastic quantities that reflect the fluctuating local environment experienced by the molecules.

In this treatment, the influence of the solvent particles on the solute, as in a solvated molecule or protein, is incorporated through additional frictional and random terms in a manner consistent with physical laws regarding equilibrium and nonequilibrium processes. Very commonly the Langevin model is applied to eliminate the explicit representation of water molecules in large and long-time simulations. Thus, applications of Langevin and Brownian dynamics simulations have been particularly successful for macroscopic models of biomolecules, such as long DNA of thousands of base pairs. Protein applications include long-timescale enzyme catalysis events.

In its simplest form, the Langevin equation in which the internal force is augmented by a frictional term proportional to the velocity, and a random force **R** which crudely mimics molecular collisions and viscosity in realistic environments, takes the form

$$m\frac{d^2}{dt^2}\mathbf{r} = -\nabla V(\mathbf{r}) - m\gamma\frac{d}{dt}\mathbf{r} + \mathbf{R}(t)\ . \tag{61}$$

Here V is the potential energy, γ is the damping constant (or collision frequency), and **R**(t) is the *white noise* vector with zero mean. It can be shown that the frictional force and the random force are related by the *fluctuation/dissipation* theorem. This relation is expressed as

$$\langle \boldsymbol{R}(t)\boldsymbol{R}(t')^T\rangle = 2\gamma k_B T m \delta(t-t')\ . \tag{62}$$

Here, equation 62 only has diagonal elements since hydrodynamic interactions between particles have been neglected. The damping constant γ controls both the magnitude of the frictional force and the variance of the random forces. It ensures that the system converges to a Boltzmann distribution characterized by the temperature T. The larger the value of γ, the greater the influence of the surrounding fluctuating force (solvent).

In the limit of small γ, the motion is termed *inertial*, and in the limit of large γ it is *diffusive* or *Brownian*. Since the number of degrees of freedom in the Langevin model is the number corresponding to the solute particles, the model is computationally much cheaper than the corresponding all-atom representation which includes explicit solvent.

In the diffusive limit, the motion is more random or *Brownian* in character. Such motion characterizes in a global sense a dense system in which the solute collides often with the surrounding fluid particles, and is thus continuously and significantly reoriented by the solvent molecules. Specifically, the Brownian regime assumes that the velocity relaxation time (γ^{-1}) is smaller than position relaxation time.

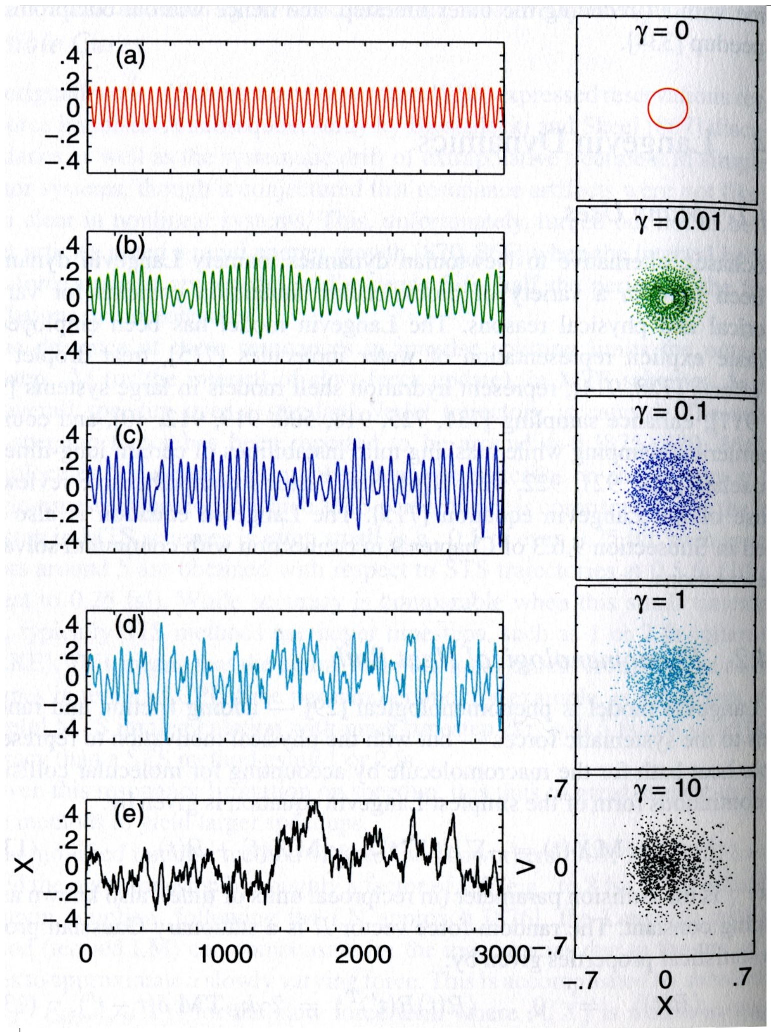


Figure II.12: Langevin trajectories for a harmonic oscillator of angular frequency ω=1 and unit mass simulated by a Verlet extension to Langevin dynamics at a timestep of 0.1 for various γ. To the right of the coordinate vs time, are phase space plots. © Springer, ref. 4.

Figure II.12 illustrates the effects of increasing γ on the trajectories and phase diagrams of a harmonic oscillator. The systematic harmonic motion and the closed, circular trajectories characteristic at zero viscosity change as the relative contribution of the random systematic forces increases. Since the stochastic Langevin forces mimic collisions between solvent molecules and the biomolecule (the solute), we see that the characteristic vibrational frequencies of a molecule in vacuum are damped. In particular, the low frequency vibrational modes are overdamped, and various correlation functions are smoothed. The magnitude of such disturbances with respect to Newtonian behavior depends on γ.

A physical value for γ for each particle can be chosen according to Stoke's law for a hydrodynamic spherical particle of radius *a*:

$$\gamma = 6\pi\eta a/m \quad , \tag{63}$$

where *m* is the particle's mass, and η is the solvent viscosity.

It is also possible to choose an approximate value for γ for a system modeled by the simple Langevin equation so as to reproduce observed experimental translation diffusion constants, *D*. Namely, in the diffusive limit *D* is related to γ by

$$D = k_B T / m\gamma \quad . \tag{64}$$

Integration of Langevin's Equations of Motion

Numerical integration of equation 61 proceeds very similar to the way described in section 3 above. A straightforward algorithm integrates the equations of motion over a time interval Δt that is sufficiently short so that the interparticle forces remain approximately constant. The algorithm for advancing the positions $r_i(t)$ is:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + c_1 \left(\frac{d\mathbf{r}}{dt}\right)_t \Delta t + c_2 \left(\frac{d^2\mathbf{r}}{dt^2}\right)_t \Delta t^2 + \Delta\mathbf{r}^G \tag{65}$$

and velocities $v_i(t)$:

$$\mathbf{v}(t+\delta t) = c_0 \left(\frac{d\mathbf{r}}{dt}\right)_t + c_1 \left(\frac{d^2\mathbf{r}}{dt^2}\right)_t \Delta t + \Delta\mathbf{v}^G \tag{66}$$

The Δr^G and Δv^G are random vectors chosen from a Gaussian distribution with zero mean and standard deviations given by:

$$\sigma_r^2 = \Delta t^2 \left(\frac{k_B T}{m}\right) \frac{1}{\gamma\Delta t} \left(2 - \frac{1}{\gamma\Delta t}(3 - 4exp(-\gamma\Delta t) + exp(-2\gamma\Delta t))\right) \tag{67}$$

and

$$\sigma_v^2 = \left(\frac{k_B T}{m}\right)(1 - exp(-\gamma\Delta t)) \; . \tag{68}$$

The numerical coefficients are given by

$$\begin{aligned} c_0 &= exp(-\gamma\Delta t) \\ c_1 &= \frac{1-c_0}{\gamma\Delta t} \\ c_2 &= \frac{1-c_1}{\gamma\Delta t} \end{aligned} \quad . \tag{69}$$

At low values of the friction coefficient, the dynamical aspects dominate. If the interparticle forces are taken to vary linearly with time between each time step, the equations of motion can be rewritten in a form that is said to produce a more accurate simulation:

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + c_1 \left(\frac{d\mathbf{r}}{dt}\right)_t \Delta t + c_2 \left(\frac{d^2\mathbf{r}}{dt^2}\right)_t \Delta t^2 + \Delta \mathbf{r}^G \tag{70}$$

And

$$\mathbf{v}(t+\delta t) = c_0 \left(\frac{d\mathbf{r}}{dt}\right)_t + (c_1 - c_2) \left(\frac{d^2\mathbf{r}}{dt^2}\right)_t \Delta t + c_2 \left(\frac{d^2\mathbf{r}}{dt^2}\right)_{t+\Delta t} \Delta t + \Delta \mathbf{v}^G \tag{71}$$

that as $\gamma \rightarrow 0$ we obtain back the velocity Verlet algorithm. For large values of γ, the random collisions dominate and the motion becomes diffusion-like.

List of references:

1. *Understanding Molecular Simulation: From Algorithms to Applications*, D. Frenkel and B. Smit, Academic Press, 2nd Edition, ISBN: 978-0-12-267351-1.
2. *Computer Simulation of Liquids*, M. P. Allen and D. J. Tildesley, Oxford.
3. *Molecular Modelling: Principles and Applications*, A. R. Leach, Prentice Hall.
4. *Molecular Modeling and Simulation: An Interdisciplinary Guide*, T. Schlick, Springer.
5. *The Art of Molecular Dynamics Simulation*, D. C. Rapaport, Cambridge University Press.
6. *Molecular Modelling for Beginners*, A. Hinchliffe, Wiley.
7. *Molecular Dynamics Simulation: Elementary Methods*, J. M. Haile, Wiley.
8. *Polymer Physics*, M. Rubinstein and R. H. Colby, Oxford.