5
Some tricks of the trade

5.1 Introduction

The purpose of this chapter is to put flesh on the bones of the techniques that have been outlined in Chapters 3 and 4. There is a considerable gulf between understanding the ideas behind the MC and MD methods, and writing and running efficient programs. In this chapter, we describe some of the programming techniques commonly used in the simulation of fluids. There are a number of similarities in the structure of MC and MD programs. They involve a start-up from an initial configuration of molecules, the generation of new configurations in a particular ensemble, and the calculation of observable properties by averaging over a finite number of configurations. Because of the similarities, most of the ideas developed in this chapter are applicable to both techniques, and we shall proceed with this in mind, pointing out any specific exceptions. The first part of this chapter describes the methods used to speed up the evaluation of the interactions between molecules, which are at the heart of a simulation program. The second part describes the overall structure of a typical program and gives details of running a simulation.

5.2 The heart of the matter

In Chapter 1, we gave an example of the calculation of the potential energy for a system of particles interacting via the pairwise Lennard-Jones potential. At that point, we paid little attention to the efficiency of that calculation, although we have mentioned points such as the need to avoid the square root function, and the relative speeds of arithmetic operations (see Chapter 1 and Appendix A). The calculation of the potential energy of a particular configuration (and, in the case of MD, the forces acting on all molecules) is the heart of a simulation program, and is executed many millions of times. Great care must be taken to make this particular section of code as efficient as possible. In this section we return to the force/energy routine with the following questions in mind. Is it possible to avoid expensive function evaluations when we calculate the forces on a molecule? What can we do with much more complicated forms of pair potential?

5.2.1 Efficient calculation of forces, energies, and pressures

Consider, initially, an atomic system with a pairwise potential $v(r)$. Assume that we have identified a pair of atoms i and j. Using the minimum image separations, the squared interatomic distance is readily calculated. The force on atom i due to j is

$$\mathbf{f}_{ij} = -\boldsymbol{\nabla}_{\mathbf{r}_i} v(r_{ij}) = -\boldsymbol{\nabla}_{\mathbf{r}_{ij}} v(r_{ij}). \tag{5.1}$$

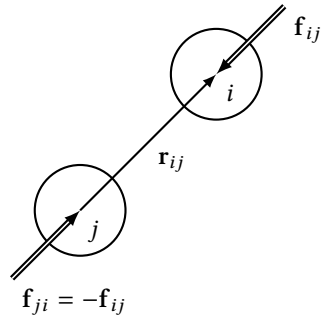


Fig. 5.1 The separation vector and force between two molecules. $\mathbf{r}_{ij}$ is the vector to i from j; $\mathbf{f}_{ij}$ is the force on i due to j; $\mathbf{f}_{ji}$ is the force on j due to i. Here the forces are drawn corresponding to an attractive interaction.

This force is directed along the interatomic vector $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ (see Fig. 5.1) and it is easy to show that

$$\mathbf{f}_{ij} = -\frac{1}{r_{ij}}\left(\frac{\mathrm{d}v(r_{ij})}{\mathrm{d}r_{ij}}\right)\mathbf{r}_{ij} = -\frac{w(r_{ij})}{r_{ij}^2}\mathbf{r}_{ij}. \tag{5.2}$$

This equation makes it clear that if $v(r_{ij})$ is an even function of r_{ij}, then the force vector can be calculated without ever working out the absolute magnitude of $\mathbf{r}_{ij}$: r_{ij}^2 will do. The function $w(r_{ij})$ is the pair virial function introduced in eqns (2.65)–(2.69). If $v(r_{ij})$ is even in r_{ij}, then so is $w(r_{ij})$. Taking the Lennard-Jones potential, eqn (1.6), as our example, we have

$$\mathbf{f}_{ij} = \frac{24\epsilon}{r_{ij}^2}\left[2\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\mathbf{r}_{ij}. \tag{5.3}$$

In an MD simulation, $v(r_{ij})$, $w(r_{ij})$, and $\mathbf{f}_{ij}$ are calculated within a double loop over all pairs i and j as outlined in Chapter 1. The force on particle j is calculated from the force on i by exploiting Newton's third law. There are one or two elementary steps that can be taken to make this calculation efficient, and these appear in Code 3.4. In an MC calculation, $v(r_{ij})$ and $w(r_{ij})$ will typically be calculated in a loop over j, with i (the particle being given a trial move) specified. This is illustrated in Code 4.3.

The calculation of the configurational energy and the force can be readily extended to molecular fluids in the interaction site formalism. In this case the potential energy is given by eqn (1.12) and the virial by (for example) eqn (2.69). If required, the forces are calculated in a straightforward way. In this case, it may be simplest to calculate the virial by using the definitions (compare eqns (2.65), (2.67))

$$w(r_{ab}) = -\mathbf{r}_{ab} \cdot \mathbf{f}_{ab}, \tag{5.4}$$

summed over all distinct site–site separations $\mathbf{r}_{ab}$ and forces $\mathbf{f}_{ab}$ (including intramolecular ones) or

$$w(r_{ij}) = -\mathbf{r}_{ij} \cdot \mathbf{f}_{ij}, \tag{5.5}$$

summed over distinct pairs of molecules, where $\mathbf{f}_{ij}$ is the sum of site–site interactions $\mathbf{f}_{ab}$ acting between each pair. These equations translate easily into code. For more complicated

intermolecular potentials, for example involving multipoles, the expressions given in Appendix C may be used.

There are some special considerations which apply to MC simulations, and which may improve the efficiency of the program. When a molecule i is subjected to a trial move, the new interactions with its neighbours j are calculated. It is possible to keep a watch for substantial overlap energies during this calculation: if one is detected, the remainder of the loop over j is immediately skipped and the move rejected. The method is particularly effective in the simulation of hard-core molecules, when a single overlap is sufficient to guarantee rejection. Note that it only makes sense to test the trial configuration in this way, since the current configuration is presumably free of substantial overlaps. For soft-core potentials, care should be taken not to set the overlap criterion at too low an energy: occasional significant overlaps may make an important contribution to some ensemble averages.

If no big overlaps are found, the result of the loops over j is a change in potential energy which is used in the MC acceptance/rejection test. If the move is accepted, this number can be used to update the current potential energy (as seen in Section 4.4): there is no need to recalculate the energy from scratch. It may be worth considering a similar approach when calculating the virial; that is, compute the change in this function which accompanies each trial move, and update $\mathcal{W}$ if it is accepted. Whether this is cost-effective compared with a less frequent complete recalculation of $\mathcal{W}$ depends on the acceptance ratio: it would not be worthwhile if a large fraction of moves were rejected. In any case, a complete recalculation of the energy and the virial should be carried out at the end of the simulation as a check that all is well.

The calculation of the pressure in systems of hard molecules (whether in MC or in MD) is carried out in a slightly different fashion, not generally within the innermost loop of the program, and we return to this in Section 5.5.

5.2.2 Table look-up and spline fit potentials

As the potentials used in simulations become more complicated, the repeated evaluation of algebraic expressions for the potential and forces can be avoided by using a prepared table. This technique has been used in the simulation of the Barker–Fisher–Watts potential for argon (Barker et al., 1971), which contains 11 adjustable parameters and an exponential.

The table is constructed once, at the beginning of the simulation program, and the potential and force are calculated as functions of $s = r_{ij}^2$. For example, we might set up a table to calculate the exponential-6 potential,

$$v^{\mathrm{E6}}(r) = -A/r^6 + B\exp(-Cr) \quad \Rightarrow \quad v^{\mathrm{E6}}(s) = -A/s^3 + B\exp(-Cs^{1/2}), \tag{5.6}$$

where A, B, and C are parameters. During the course of the run, values of $s = r_{ij}^2$ are calculated for a particular pair of molecules, and the potential is interpolated from the table. Polynomial or rational function interpolation from a set of tabulated values and methods for efficiently searching an ordered table are discussed by Press et al. (2007, Chapter 3). In a molecular dynamics simulation, of course, we also need to evaluate the forces. We may compute these by constructing a separate table of values of the function $w(r_{ij})/r_{ij}^2$, which enters into the force calculation through eqn (5.2). The success of the interpolation method depends on the careful choice of the table-spacing. Typically, we

find that $\delta s = \delta r_{ij}^2 = 0.01 r_{\mathrm{m}}^2$, where r_{m} is the position of the potential minimum, produces a sufficiently fine grid for use in MD and MC simulations.

Andrea et al. (1983) suggested an improvement to this method, using a spline fit to the potential. The function $v(s)$ (where once more $s = r_{ij}^2$) is divided into a number of regions by grid points or knots s_k. In each interval (s_k, s_{k+1}) the function is approximated by a fifth-order polynomial

$$v(s) \approx \sum_{n=0}^{5} c_n^{(k)} (s - s_k)^n. \tag{5.7}$$

The coefficients $c_0^{(k)} \cdots c_5^{(k)}$ are uniquely determined by the exact values of $v(s)$, $\mathrm{d}v(s)/\mathrm{d}s$, and $\mathrm{d}^2 v(s)/\mathrm{d}s^2$ at the two ends of the interval (Andrea et al., 1983, Appendix). Thus, we need to store the grid points s_k (which need not be evenly spaced) and six coefficients for each interval. In their simulation of water, Andrea et al. represented the O–O, O–H, and H–H potentials using 14, 16, and 26 intervals respectively. For MD, the forces are easily obtained by differentiating eqn (5.7) and using

$$\frac{w(r_{ij}^2)}{r_{ij}^2} = \frac{w(s)}{s} = 2\frac{\mathrm{d}v}{\mathrm{d}s}. \tag{5.8}$$

Another interesting example of a force represented by a polynomial fit is the construction of the force matched (FM) representation of a spherically truncated Coulomb interaction as discussed in Section 6.4 (Izvekov et al., 2008). In this case the truncated Coulomb force, $q_i q_j / s$ for $s^{1/2} < r_{\mathrm{c}}$, is represented by the polynomial

$$f_{ij}^{\mathrm{FM}}(s) = q_i q_j \left[\frac{1}{s} + \sum_{k=0}^{7} a_k s^{k/2} \right], \quad r_{\mathrm{core}} < s^{1/2} < r_{\mathrm{c}}, \tag{5.9}$$

where the coefficients a_k are determined by matching the trajectories from the FM force and the true long-range Coulomb interaction.

5.2.3 Shifted and shifted-force potentials

The truncation of the intermolecular potential at a cutoff introduces some difficulties in defining a consistent potential and force for use in the MD method. The function $v(r_{ij})$ used in a simulation contains a discontinuity at $r_{ij} = r_{\mathrm{c}}$: whenever a pair of molecules crosses this boundary, the total energy will not be conserved. We can avoid this by shifting the potential function by an amount $v_{\mathrm{c}} = v(r_{\mathrm{c}})$, that is, using instead the function

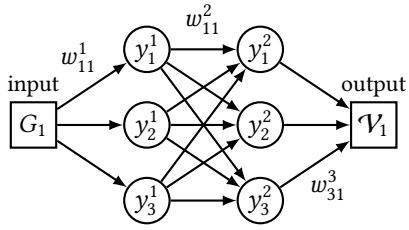
$$v^{\mathrm{S}}(r_{ij}) = \begin{cases} v(r_{ij}) - v_{\mathrm{c}} & r_{ij} \le r_{\mathrm{c}} \\ 0 & r_{ij} > r_{\mathrm{c}}. \end{cases} \tag{5.10}$$

The small additional term is constant for any pair interaction, and does not affect the forces, and hence the equations of motion of the system. However, its contribution to the total energy varies from timestep to timestep, since the total number of pairs within cutoff range varies. This term should certainly be included in the calculation of the total energy, so as to check the conservation law. However, there is a further problem. The

Example 5.1 Learning the potentials with a neural network

For complicated intermolecular potentials, such as those developed from *ab initio* calculations, it can be cost-effective to teach the computer to estimate the potential energy of a particular atom from a knowledge of its environment in the liquid. This can be done by constructing an artificial neural network (Behler, 2011).

In this approach, the Cartesian coordinates of a particular atom, i, are used to calculate a set of μ symmetry functions, G_i^μ. The first of these functions, $\mu = 1$, might depend on the radial distribution of all the other atoms around i. The second might depend on the distribution of triplets around atom i through the variable $\cos\theta_{ijk}$, and so on (see e.g. Behler and Parrinello, 2007, eqns (4) and (5)). Symmetry functions of different order can be combined with weights $w_{i\mu}^s$ to produce an overall $G_i = \sum_\mu w_{i\mu}^s G_i^\mu$. The G_i are the input to a simple neural network of the kind shown.



There is a separate network for each atom, containing a number of hidden layers (two in this example) each with a number of nodes (three in this example). The value y_m^ℓ held by the network at a particular node, m, in layer, ℓ, is calculated as a function of the weighted sum of the connected nodes in the previous layer, that is, $y_m^\ell = f(\sum_{n=1}^3 w_{nm}^\ell y_n^{\ell-1})$ for $\ell = 2$ in our example. The activation function is often taken to be $f(x) = \tanh(x)$, although different functions can be used for different layers. The output from the network is the potential energy, $\mathcal{V}_1$. The outputs from all the separate networks can be added to produce the total potential energy, $\mathcal{V}_{\text{net}}$. For a number of training configurations the full $\mathcal{V}_{\text{QM}}$ is also calculated. The weights, w, in the network, and symmetry functions, are adjusted to minimize $|\mathcal{V}_{\text{net}} - \mathcal{V}_{\text{QM}}|$ for the training set. The same connectivity and inter-layer weights are used in the different networks established for each atom. Once the optimum weights have been established, the network can be used to accurately estimate the potential and forces for an unknown configuration.

Morawietz et al. (2016) have employed neural network potentials in *ab initio* molecular dynamics simulations of water. The efficiency of the approach enabled them to show that the relatively weak, isotropic van der Waals forces are crucial in producing the density maximum and the negative volume of melting of the fluid.

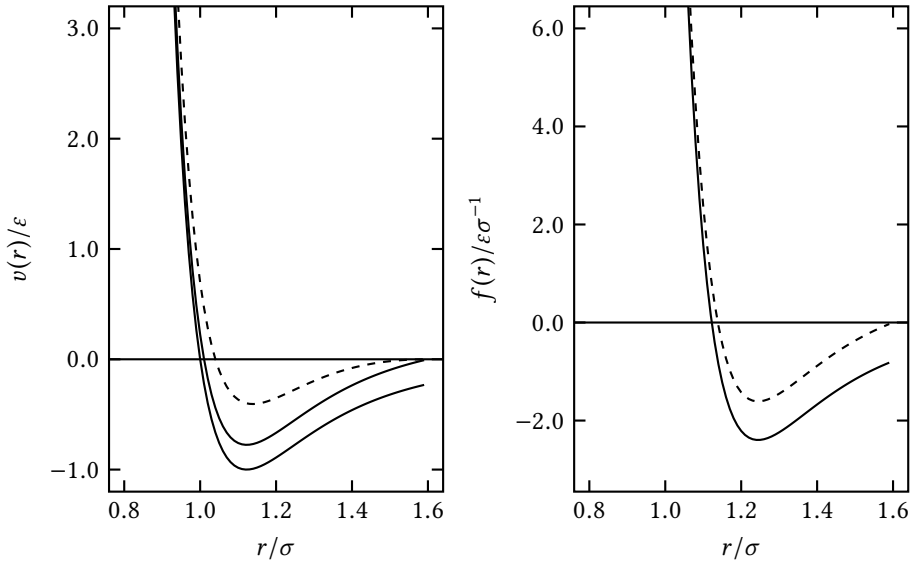


Fig. 5.2 Magnitude of the pair potential $v(r)$ and the force $f(r)$, for the Lennard-Jones and shifted Lennard-Jones potentials (solid lines), and the shifted-force modification (dashed lines). Note that, for illustration, we have chosen a very short cutoff distance, $r_c = 1.6\sigma$.

force between a pair of molecules is still discontinuous at $r_{ij} = r_c$. For example, in the Lennard-Jones case, the force is given by eqn (5.3) for $r_{ij} \leq r_c$, but is zero for $r_{ij} > r_c$. The magnitude of the discontinuity is $\approx 0.039\,\epsilon\sigma^{-1}$ for $r_c = 2.5\sigma$. It can cause instability in the numerical solution of the differential equations. To avoid this difficulty, a number of workers have used a 'shifted-force potential' (Stoddard and Ford, 1973; Streett et al., 1978; Nicolas et al., 1979; Powles et al., 1982). A small linear term is added to the potential, so that its derivative is zero at the cutoff distance

$$v^{\mathrm{SF}}(r_{ij}) = \begin{cases} v(r_{ij}) - v_c - (r_{ij} - r_c)\left(\dfrac{\mathrm{d}v(r_{ij})}{\mathrm{d}r_{ij}}\right)_{r_{ij}=r_c} & r_{ij} \leq r_c \\ 0 & r_{ij} > r_c. \end{cases} \tag{5.11}$$

The discontinuity now appears in the gradient of the force, not in the force itself. The shifted-force potential for the Lennard-Jones case is shown in Fig. 5.2. The force goes smoothly to zero at the cutoff r_c, removing problems in energy conservation and any numerical instability in the equations of motion. Making the additional term quadratic (Stoddard and Ford, 1973) avoids taking a square root. Of course, the difference between the shifted-force potential and the original potential means that the simulation no longer corresponds to the desired model liquid. However, the thermodynamic properties of a fluid of particles interacting with the unshifted potential can be recovered from the shifted-force potential simulation results, using a simple perturbation scheme (Nicolas et al., 1979; Powles, 1984).

5.2.4 Thermodynamic properties with a truncated potential

Comparing simulation results obtained with different versions of a potential can lead to considerable confusion. Here we discuss the most common situations. First, suppose that an MC simulation of a model fluid has been performed with a truncated (but not shifted) pair potential, which we write formally as

$$v_{\mathrm{c}}(r) = v(r)\Theta(r_{\mathrm{c}} - r) \tag{5.12}$$

where r_{c} is the cutoff distance and Θ is the unit step function. The corresponding virial function for the calculation of the pressure is

$$w_{\mathrm{c}}(r) = w(r)\Theta(r_{\mathrm{c}} - r) - rv(r)\delta(r - r_{\mathrm{c}}). \tag{5.13}$$

The second term arises from the discontinuity of the potential at the cutoff. Using eqn (2.102)

$$\left\langle \sum_i \sum_{j>i} a(r_{ij}) \right\rangle = \tfrac{1}{2} N\rho \int_0^\infty a(r) g(r) 4\pi r^2 \, \mathrm{d}r. \tag{5.14}$$

and recalling $\mathcal{W} = -\frac{1}{3}\sum_i \sum_{j>i} w(r_{ij})$, we can evaluate the average of the delta function in calculating the pressure

$$P_{\mathrm{c}}V = Nk_{\mathrm{B}}T + \langle \mathcal{W} \rangle + \frac{2\pi N\rho}{3} r_{\mathrm{c}}^3 g(r_{\mathrm{c}}) v(r_{\mathrm{c}}) \approx Nk_{\mathrm{B}}T + \langle \mathcal{W} \rangle + \underbrace{\frac{2\pi N\rho}{3} r_{\mathrm{c}}^3 v(r_{\mathrm{c}})}_{\Delta P_{\mathrm{c}} V}, \tag{5.15}$$

thereby defining the correction ΔP_{c} for the delta function contribution. The approximation results from assuming that the cutoff is large enough that $g(r_{\mathrm{c}}) \approx 1$. It is understood that $\mathcal{W}$ involves the sum over distinct pairs of the first term on the RHS of eqn (5.13), that is, those pairs within the cutoff. Each of these three terms contributes to the pressure associated with the truncated potential, hence the notation P_{c}. The value of ΔP_{c} can be calculated in advance of the simulation. For the Lennard-Jones potential, in reduced units, it is

$$\Delta P_{\mathrm{c}}^* = \tfrac{8}{3}\pi\rho^{*2}\left(r_{\mathrm{c}}^{*-9} - r_{\mathrm{c}}^{*-3}\right). \tag{5.16}$$

Although the truncated potential is used to sample the canonical ensemble, it is the pressure corresponding to the full, untruncated, potential that is normally of interest. Now, the quantity to be averaged is the pairwise sum of $w(r)$ functions, that is, it involves no delta functions. Moreover, in a perturbative, mean-field, approach, the long-range term in the potential is assumed to have no effect on the structure. Therefore the pressure of the full system can be written, assuming $g(r) = 1$ for $r > r_{\mathrm{c}}$

$$PV \approx Nk_{\mathrm{B}}T + \langle \mathcal{W} \rangle - \frac{2\pi}{3} N\rho \int_{r_{\mathrm{c}}}^\infty w(r) r^2 \, \mathrm{d}r \equiv Nk_{\mathrm{B}}T + \langle \mathcal{W} \rangle + P_{\mathrm{LRC}} V. \tag{5.17}$$

$\mathcal{W}$ here has exactly the same interpretation as in eqn (5.15); that is, the sum of the virial functions of distinct pairs within cutoff range, measured in the simulation using the truncated potential. The delta function correction does not appear at all, and eqn (5.17)

can be used straightforwardly to compute the pressure during the simulation. The long-range correction, P_{LRC}, can be computed ahead of the simulation. For the example of the Lennard-Jones potential, it is given in eqn (2.144)

$$P^*_{\mathrm{LRC}} = \tfrac{16}{9}\pi\rho^{*2}\left(2r_{\mathrm{c}}^{*-9} - 3r_{\mathrm{c}}^{*-3}\right). \tag{5.18}$$

It may be desirable to measure the pressure P_{c} corresponding to the actual truncated potential used in the simulation, and express the pressure for the full potential in terms of this. In this case, the relevant formula is obtained by combining eqns (5.15) and (5.17):

$$P = P_{\mathrm{c}} - \Delta P_{\mathrm{c}} + P_{\mathrm{LRC}}. \tag{5.19}$$

In other words, the delta function correction used in the calculation of P_{c} must be subtracted off again, and the long-range correction added on. This is a very common approach to use in constant-pressure Monte Carlo simulations of the truncated potential, because then the specified input pressure corresponds to P_{c}, not P. The average of P_{c} may be calculated (as a consistency check) in the simulation, using eqn (5.15), but in any case the measured average density ρ will correspond to an equation of state $\rho(P_{\mathrm{c}})$, which we would normally invert to get $P_{\mathrm{c}}(\rho)$. We can recover $P(\rho)$ afterwards using eqn (5.19). For the Lennard-Jones potential, the *combined* correction term is

$$P^* - P^*_{\mathrm{c}} = P^*_{\mathrm{LRC}} - \Delta P^*_{\mathrm{c}} = \pi\rho^{*2}\left(\tfrac{8}{9}r_{\mathrm{c}}^{*-9} - \tfrac{8}{3}r_{\mathrm{c}}^{*-3}\right). \tag{5.20}$$

This formula was used by Finn and Monson (1989) and Smit (1992), omitting the r_{c}^{*-9} term which is relatively small, to compare results from the truncated Lennard-Jones potential with the equation of state of Nicolas et al. (1979).

A little thought reveals that the same result will be obtained (on the grounds of ensemble equivalence) if one uses eqn (5.17) directly, even in a constant-pressure simulation of the truncated potential. However, one must be aware that the value of P obtained from it will not agree with the specified pressure of the simulation. In this case, the long-range correction term will vary with density.

Corresponding molecular dynamics simulations are performed with a truncated pair force

$$f_{\mathrm{cs}}(r) = f(r)\Theta(r_{\mathrm{c}} - r), \tag{5.21}$$

where the notation stands for 'cut and shifted'. In this case the virial function inside the cutoff is the same as that of the full potential, but there is no discontinuity at $r = r_{\mathrm{c}}$. Therefore there is no extra term in the calculation of the pressure for this model

$$P_{\mathrm{cs}}V = Nk_{\mathrm{B}}T + \langle \mathcal{W} \rangle. \tag{5.22}$$

Making the same assumptions as before, the pressure for the full potential is straightforwardly given by the long-range correction

$$P = P_{\mathrm{cs}} + P_{\mathrm{LRC}}. \tag{5.23}$$

The corresponding potential energy function, calculated by integrating the negative force from infinity to r is

$$v_{\mathrm{cs}}(r) = \left[v(r) - v(r_{\mathrm{c}})\right]\Theta(r_{\mathrm{c}} - r), \tag{5.24}$$

and the total potential energy is

$$\mathcal{V}_{\rm cs} = \sum_i \sum_{j>i} v_{\rm cs}(r_{ij}) = \mathcal{V} - N_{\rm c} v(r_{\rm c}), \tag{5.25}$$

where $\mathcal{V}$ is the total potential energy for all pairs within cutoff range (without shifting), and $N_{\rm c}$ is the total number of such pairs

$$N_{\rm c} = \sum_i \sum_{j>i} \Theta(r_{\rm c} - r_{ij}). \tag{5.26}$$

$\mathcal{V}_{\rm cs}$ is the true potential energy for the truncated force and it is this potential that must be added to the kinetic energy to check for total energy conservation in the simulation. In estimating the potential energy for the full Lennard-Jones fluid, it is consistent to calculate the potential inside the cutoff using just the Lennard-Jones term without the shift and to add on the mean-field, long-range correction given by eqn (2.143).

Monte Carlo calculations may also be performed using the 'cut-and-shifted' potential, and the pressure for the full potential estimated using the long-range correction $P_{\rm LRC}$. The discontinuity term $\Delta P_{\rm c}$ does not arise in this case.

5.3 Neighbour lists

In the inner loops of the MD and MC programs, we consider a molecule i and loop over all molecules j to calculate the minimum image separations. If molecules are separated by distances greater than the potential cutoff, the program skips to the end of the inner loop, avoiding expensive calculations, and considers the next neighbour. In this method, the time to examine all pair separations is proportional to N^2. Verlet (1967) suggested a technique for improving the speed of a program by maintaining a list of the neighbours of a particular molecule, which is updated at intervals. Between updates of the neighbour list, the program does not check through all the j molecules, but just those appearing on the list. The number of pair separations explicitly considered is reduced. This saves time in looping through j, minimum imaging, calculating r_{ij}^2, and checking against the cutoff, for all those particles not on the list. Obviously, there is no change in the time actually spent calculating the energy and forces arising from neighbours within the potential cutoff. In this section, we describe some useful time-saving neighbour list methods. These methods are equally applicable to MC and MD simulations, and for convenience we use the MD method to illustrate them. There are differences concerning the relative sizes of the neighbour lists required in MC and MD and we return to this point at the end of the next section. Related techniques may be used to speed up MD of hard systems (Erpenbeck and Wood, 1977). In this case, the aim is to construct and maintain, as efficiently as possible, a table of future collisions between pairs of molecules. The scheduling of molecular collisions has been discussed in detail by Rapaport (1980) and Bannerman et al. (2011).

5.3.1 The Verlet neighbour list

In the original Verlet method, the potential cutoff sphere, of radius $r_{\rm c}$, around a particular molecule is surrounded by a 'skin' to give a larger sphere of radius r_ℓ, as shown in Fig. 5.3. At the first step in a simulation, a list is constructed of all the neighbours of each molecule,

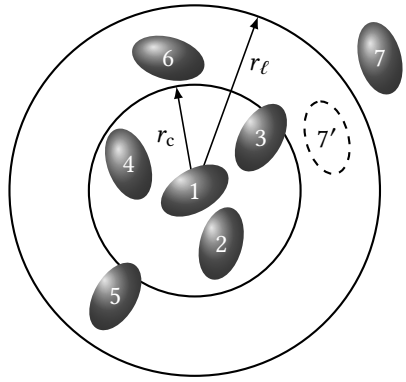


Fig. 5.3 The cutoff sphere, radius r_c, and its skin, radius r_ℓ, around a molecule 1. Molecules 2, 3, 4, 5, and 6 are on the list of molecule 1; molecule 7 is not. Only molecules 2, 3, and 4 are within the range of the potential at the time the list is constructed.

Code 5.1 Force routine using Verlet neighbour list

These files are provided online. `md_lj_vl_module.f90` contains a Lennard-Jones force routine using a Verlet neighbour list, while `verlet_list_module.f90` contains the routines for constructing and updating the list. These two files together act as a drop-in replacement for `md_lj_module.f90` (see Code 3.4). They can be combined with, for example, `md_nve_lj.f90` of Code 3.4 and the utility modules described in Appendix A to make a molecular dynamics program, as illustrated in the supplied `SConstruct` file.

```
! md_lj_vl_module.f90
! Force routine for MD, LJ atoms, Verlet neighbour list
MODULE md_module

! verlet_list_module.f90
! Verlet list handling routines for MD simulation
MODULE verlet_list_module
```

for which the pair separation is within r_ℓ. These neighbours are stored in an array, called, shall we say, `list`. `list` is quite large, of dimension roughly $4\pi r_\ell^3 \rho N/6$. At the same time, a second indexing array, `point`, of size N, is constructed. `point(i)` points to the position in the array `list` where the first neighbour of molecule `i` can be found. Since `point(i+1)` points to the first neighbour of molecule `i+1`, then `point(i+1)-1` points to the last neighbour of molecule `i`. Thus, using `point`, we can readily identify the part of the large `list` array which contains neighbours of `i`. Routines for setting up the arrays `list` and `point` are given in Code 5.1.

Over the next few timesteps, the list is used in the force/energy evaluation routine. For each molecule `i`, the program identifies the neighbours `j`, by running over `list` from `point(i)` to `point(i+1)-1`. This loop should automatically handle the case when molecule

`i` has no neighbours, and can be skipped, in which case `point(i+1)-1` will be less than `point(i)`. This is certainly possible in dilute systems. A sample force routine using the Verlet list is given in Code 5.1. From time to time, the neighbour list is reconstructed, and the cycle is repeated. The algorithm is successful because the skin around r_c is chosen to be thick enough so that between reconstructions a molecule, such as 7 in Fig. 5.3, which is not on the list of molecule 1, cannot penetrate through the skin into the important r_c sphere. Molecules such as 3 and 4 can move in and out of this sphere, but since they are on the list of molecule 1, they are always considered regardless, until the list is next updated.

The interval between updates of the table is often fixed at the beginning of the program, and intervals of 10–20 steps are quite common. An important refinement allows the program to update the neighbour list automatically. When the list is constructed, a vector for each molecule is set to zero. At subsequent steps, the vector is incremented with the displacement of each molecule. Thus it stores the total displacement for each molecule since the last update. When the sum of the magnitudes of the two largest displacements exceeds $r_\ell - r_c$, the neighbour list should be updated again (Fincham, 1981; Thompson, 1983). The code for automatic updating of the neighbour list is given in the routine of Code 5.1.

The list sphere radius, r_ℓ, is a parameter that we are free to choose. As r_ℓ is increased, the frequency of updates of the neighbour list will decrease. However, with a large list, the efficiency of the non-update steps will decrease. This trade-off is illustrated in Fig. 5.4, for different system sizes. For systems of a few hundred particles, the initial improvement reflects the interval between updates, which is when an all-pairs calculation is needed. The actual cost of using the list is relatively small, and a significant speed increase is seen up to $r_\ell^* \approx 2.8$. For larger values of r_ℓ^*, the cost of looping over all particles within the list becomes more significant. For systems $N > 500$, the improvement is dramatic and the turnover point shifts to larger r_ℓ^*, but as we shall see, the method of Section 5.3.2 becomes preferable. As the size of the system becomes larger, the size of the `list` array grows, approximately $\propto N$. If storage is a priority, then a binary representation of the list can be employed (O'Shea, 1983).

In the MC method, the array `point` has a size $N + 1$ rather than N, since the index `i` runs over all N atoms rather than $N - 1$ as in MD. In addition, the array `list` is roughly twice as large in MC as in a corresponding MD program. In the MD technique, the list for a particular molecule i contains only the molecules j with an index greater than i, since in this method we use Newton's third law to calculate the force on j from i at the same time as the force on i from j. In the MC method, particles i and j are moved independently and the list must contain separately the information that i is a neighbour of j and j a neighbour of i.

5.3.2 Cell structures and linked lists

As the size of the system increases towards 1000 molecules, the conventional neighbour list becomes too large to store easily, and the logical testing of every pair in the system is inefficient. An alternative method of keeping track of neighbours for large systems is the cell index method (Quentrec and Brot, 1973; Hockney and Eastwood, 1988). The cubic simulation box (extension to non-cubic cases is possible) is divided into a regular lattice of

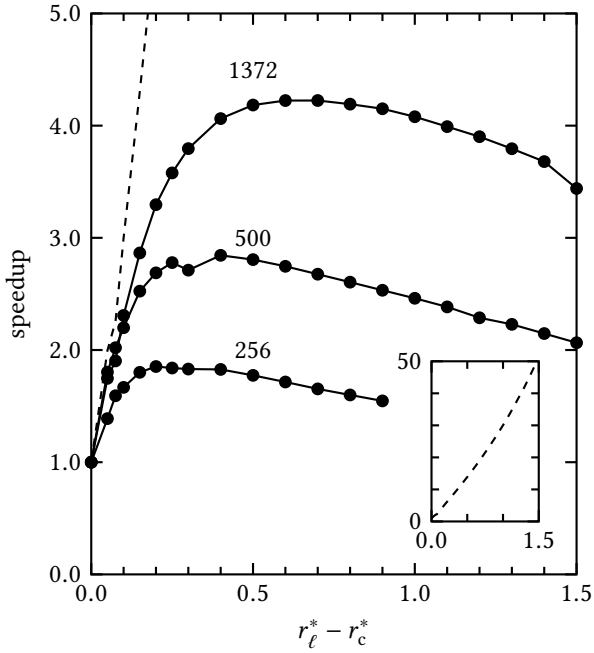


Fig. 5.4 Speedup with Verlet list. Example results are shown for the Lennard-Jones potential, with cutoff $r_c^* = 2.5$, and various values of skin thickness $r_\ell^* - r_c^*$, for the indicated system sizes N. The state point is $\rho^* = 0.78$, $T^* = 0.85$, and the timestep is $\delta t^* = 0.005$. The curves show timesteps per CPU-second, normalized by the speed for zero skin thickness (when the list is updated every step). The dashed line (also shown in the inset) gives the average number of steps between updates, which is almost independent of system size.

$s_c \times s_c \times s_c$ cells. A two-dimensional representation of this is shown in Fig. 5.5. These cells are chosen so that the side of the cell $\ell = L/s_c$ is greater than the cutoff distance for the forces, r_c. For the two-dimensional example of Fig. 5.5, the neighbours of any molecule in cell 8 are to be found in the cells 2, 3, 4, 7, 8, 9, 12, 13, and 14. If there is a separate list of molecules in each of those cells then searching through the neighbours is a rapid process. For the two-dimensional system illustrated, there are approximately $\rho\ell^2$ molecules in each cell (where ρ is the number density per unit area); the analogous result in three dimensions would be $\rho\ell^3$, where $\rho = N/V$. Using the cell structure in two dimensions, we need only examine $9N\rho\ell^2$ pairs (or just $4.5N\rho\ell^2$ if we take advantage of the third law in the MD method). This contrasts with N^2 (or $\frac{1}{2}N(N-1)$) for the brute-force approach. When the cell structure is used in three dimensions, then we compute $27N\rho\ell^3$ interactions ($13.5N\rho\ell^3$ for MD) as compared with N^2 (or $\frac{1}{2}N(N-1)$). The cost of the method scales with N rather than N^2, and the speedup over the brute-force approach is $\approx L^2/9\ell^2$ in 2D, or $\approx L^3/27\ell^3$ in 3D.

The cell structure may be set up and used by the method of linked lists (Knuth, 1973, Chapter 2; Hockney and Eastwood, 1988, Chapter 8). The first part of the method involves sorting all the molecules into their appropriate cells. This sorting is rapid, and may be

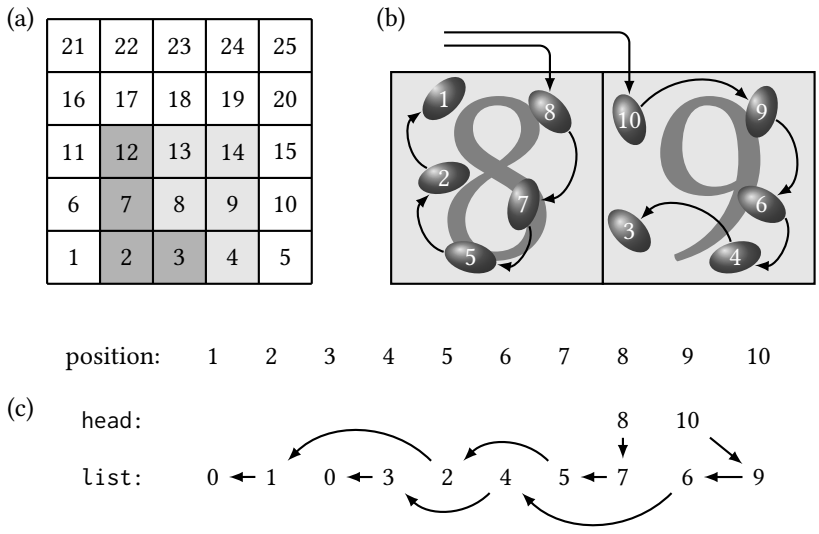


Fig. 5.5 The cell method in two dimensions. (a) The central box is divided into $s_c \times s_c$ cells ($s_c = 5$). A molecule in cell 8 may interact with molecules in any of the shaded cells. In an MD program, only the light-shaded neighbours of cell 8 need be examined in the inner loop over cells. (b) A close-up of cells 8 and 9, showing the molecules and the link-list structure. (c) The `head` and `list` array elements corresponding to these two cells. Each entry gives the position of the next in the `list` array.

performed every step. Two arrays are created during the sorting process. The 'head-of-chain' array (`head`) has one element for each cell. This element contains the identification number of one of the molecules sorted into that cell. This number is used to address the element of a linked-list array (`list`), which contains the number of the next molecule in that cell. In turn, the `list` array element for that molecule is the index of the next molecule in the cell, and so on. If we follow the trail of linked-list references, we will eventually reach an element of `list` which is zero. This indicates that there are no more molecules in that cell, and we move on to the head-of-chain molecule for the next cell. To illustrate this, imagine a simulation of particles in two cells: 1, 2, 5, 7, and 8 in cell 8 and 3, 4, 6, 9, and 10 in cell 9 (see Fig. 5.5). The `head` and `list` arrays are illustrated in the figure. For cell 8, `head(8) = 8` (the last particle found to lie in that cell), while for cell 9, `head(9) = 10`; in both cases the route through the linked list is arrowed. The construction of `head` and `list` is straightforward, and is illustrated in Code 5.2.

In MD, the calculation of the forces is performed by looping over all cells. For a given cell, the program traverses the linked list. A particular molecule on the list may interact with all molecules in the same cell that are further down the list. This avoids counting the *ij* interaction twice. A particular molecule also may interact with the molecules in the neighbouring cells. To avoid double counting of these forces, only a limited number of the neighbouring cells are considered. This idea is most easily explained with reference to our two-dimensional example shown in Fig. 5.5. A molecule in cell 8 interacts with other molecules in the same cell, and with eight neighbouring cells, but the program only

Code 5.2 Building a cell structure with linked lists

These are the essential statements for assigning each atom to a cell, and building the links between atoms lying in each cell. In the usual unit cube simulation box, the cell length is $1/s_c$. Here, we identify each cell by a three-dimensional index denoting its position in space, rather than a single index as in Fig. 5.5. This means that the cell `c(1:3,i)`, in which an atom `i` lies, is essentially a discretized form of the coordinate vector `r(1:3,i)`; it is convenient to make the indices lie in the range `(0,sc-1)` rather than `(1,sc)`. We assume that periodic boundary corrections have already been applied to `r`. Correspondingly, the `head` array is of rank 3, and each element contains the particle index `i` of the last particle added to each cell. An expanded version of the code, including guards against roundoff error, appears in the online file `link_list_module.f90` of Code 5.3.

```
REAL,    DIMENSION(3,n)                   :: r
INTEGER, DIMENSION(3,n)                   :: c
INTEGER, DIMENSION(n)                     :: list
INTEGER, DIMENSION(0:sc-1,0:sc-1,0:sc-1) :: head

head(:,:,:) = 0
DO i = 1, n
  c(:,i)  = FLOOR ( ( r(:,i) + 0.5 ) * REAL(sc) )
  list(i) = head(c(1,i),c(2,i),c(3,i))
  head(c(1,i),c(2,i),c(3,i)) = i
END DO
```

checks cells 8, 4, 9, 13, and 14. Interactions between cells 7 and 8 are checked when cell 7 is the focus of attention, and so on. In this way, we can make full use of Newton's third law in calculating the forces. We note that for a cell at the edge of the basic simulation box (15 for example in Fig. 5.5) it is necessary to consider the periodic cells (6, 11, and 16). An example of the code for constructing and searching through cells is given in Code 5.3. The linked-list method can also be used with the MC technique. In this case, a molecule move involves checking molecules in the same cell, and in all the neighbouring cells. This is illustrated in Code 5.4.

The cell structure may be used somewhat more efficiently, by avoiding unnecessary distance calculations, if the molecules in each cell are sorted into order of increasing (say) x-coordinate (Hockney and Eastwood, 1988). Extending this idea, Gonnet (2007) suggests sorting particles according to their projection onto the vector that connects the cell centres. For each pair of cells, the loops over particle pairs may then be restricted to those whose projected separation (a lower bound on the true separation) is less than r_c. Similar improvements are reviewed by Heinz and Hünenberger (2004) and Welling and Germano (2011).

The linked-list method has been used with considerable success in simulation of systems such as plasmas, galaxies, and ionic crystals, which require a large number of

Code 5.3 Force routine using linked lists

These files are provided online. `md_lj_ll_module.f90` contains a Lennard-Jones force routine using a linked list, while `link_list_module.f90` contains the routines for constructing and updating the list. These two files together act as a drop-in replacement for `md_lj_module.f90` (see Code 3.4). They can be combined with, for example, `md_nve_lj.f90` of Code 3.4 and the utility modules described in Appendix A, to make a molecular dynamics program, as illustrated in the supplied `SConstruct` file.

```
! md_lj_ll_module.f90
! Force routine for MD simulation, LJ atoms, linked lists
MODULE md_module

! link_list_module.f90
! Link list handling routines for MC or MD simulation
MODULE link_list_module
```

Code 5.4 Monte Carlo routines using linked lists

This file is provided online. `mc_lj_ll_module.f90` contains Lennard-Jones energy routines using a linked list. Together with `link_list_module.f90` (Code 5.3) it acts as a drop-in replacement for `mc_lj_module.f90` (see Code 4.3). They can be combined with several of the supplied Lennard-Jones Monte Carlo programs in the common ensembles, for example `mc_nvt_lj.f90` (Code 4.3), `mc_zvt_lj.f90` (Code 4.6), plus the utility modules described in Appendix A, to make complete MC programs, as illustrated in the `SConstruct` file.

```
! mc_lj_ll_module.f90
! Energy and move routines for MC, LJ potential, linked lists
MODULE mc_module
```

particles (Hockney and Eastwood, 1988). In both the linked-list methods and the Verlet neighbour list, the computing time required to run a simulation tends to increase linearly with the number of particles N rather than quadratically. This rule of thumb does not take into account the extra time required to set up and manipulate the lists. For small systems ($N \approx 100$) the overheads involved make the use of lists unprofitable. Mazzeo et al. (2010) have developed a linked-list method for Monte Carlo simulations, and discuss several optimization strategies.

Parallel computers offer a more cost-effective route to high-performance computing than traditional single processor machines. Both the force calculation and integration steps of molecular dynamics are parallel in nature, and for that reason parallel algorithms based on the cell-structure, linked-list technique have been developed. This method is particularly efficient when the range of intermolecular potential is much smaller than the

dimensions of the simulation box. This domain-decomposition approach was originally tested for fluids of up to 2×10^6 atoms in two and three dimensions (Pinches et al., 1991). A detailed implementation using the message passing interface (MPI) is provided by Griebel et al. (2007, Chapter 4).

One is not restricted to using cells of length r_c. A modification of this general approach employs cells that are sufficiently small that at most one particle can occupy each cell. In this case, a linked-list structure as described earlier is not required: the program simply loops over all cells, and conducts an inner loop over the cells that are within a distance r_c of the cell of interest. Advantages of this method are that the list of cells 'within range' of each given cell may be computed at the start of the program, remaining unaltered throughout, and that a simple decision (is the cell occupied or not?) is required at each stage. However, the number of cells is large, and, of course, many of them are empty. This version of the method has been analysed in detail (Mattson and Rice, 1999; Yao et al., 2004; Heinz and Hünenberger, 2004).

5.4 Non-bonded interactions and multiple timesteps

In Section 3.5 we introduced the idea of separating the Liouville operator into parts that vary at different rates in time, due to a division into 'fast' and 'slow' forces. One obvious such division is between intramolecular forces, especially bond stretching potentials which produce rapid vibration, and intermolecular non-bonded forces. This idea may be extended, and molecular dynamics packages quite commonly divide non-bonded potentials into a hierarchy of terms, from short-ranged contributions which have high gradients, to longer-ranged, more gently varying, terms. These are then handled by a corresponding hierarchy of timesteps. The rationale is that the number of long-range pair interactions is much larger than the number of short-range ones (there is a geometrical factor of r_{ij}^2) but they may be computed much less frequently.

A simplified example, following Procacci and Marchi (1996), illustrates the idea. Consider sub-dividing the Lennard-Jones potential according to a set of cutoff distances $R_1 < R_2 < \cdots < R_k < \cdots < R_K$ as follows

$$v_k^{\mathrm{LJ}}(r) \equiv \left[S_k\left(\frac{r - R_k}{\lambda}\right) - S_{k-1}\left(\frac{r - R_{k-1}}{\lambda}\right)\right] v^{\mathrm{LJ}}(r) \tag{5.27}$$

where the switching functions are defined $S_0(x) = 0$, $S_K(x) = 1$, and

$$S_k(x) = \begin{cases} 1 & x < -1 \\ (2x+3)x^2 & -1 < x < 0 \\ 0 & x > 0 \end{cases} \qquad k = 1 \ldots K-1.$$

It is understood that $R_0 = 0$, and R_K is the potential cutoff distance. The parameter λ defines a range of r over which S_k changes smoothly from 1 to 0, and the successive shells defined by the square brackets in eqn (5.27) overlap by this distance. It is easy to show that

$$v^{\mathrm{LJ}}(r) = \sum_{k=1}^{K} v_k^{\mathrm{LJ}}(r).$$

Code 5.5 Multiple-timestep algorithm, Lennard-Jones atoms

These files are provided online. `md_lj_mts.f90` carries out MD using a three-timestep scheme, for three shells of pair interaction, which are computed using the force routine supplied in `md_lj_mts_module.f90`. Utility module routines described in Appendix A handle input/output and averages. The example is slightly contrived: for clarity, no neighbour lists are used.

```
! md_lj_mts.f90
! Molecular dynamics, NVE, multiple timesteps
PROGRAM md_lj_mts

! md_lj_mts_module.f90
! Force routine for MD, LJ atoms, multiple time steps
MODULE md_module
```

Then, for a suitable choice of the R_k, each contribution k may be treated using a different timestep, using a scheme similar to that described in Section 3.5, taking advantage of the slower variation in time of the longer-range contributions. Of course, as the atoms move around, the pairs belonging to each shell will change. Also, in calculating the forces from each separate contribution $v_k^{\mathrm{LJ}}(r)$, the r-dependence of the switching functions must be taken into account. Code 5.5 illustrates this approach.

It should be emphasized that this example is oversimplified, and very little improvement in efficiency should be expected for a relatively short-ranged potential such as Lennard-Jones. The method comes into its own when electrostatic forces are present, for which the long-range parts may be handled with a long timestep, and when high-frequency intramolecular terms may be tackled with a short timestep, as discussed in Section 3.5. Procacci and Marchi (1996) propose a separation based not on atom pairs, but on pairs of neutral groups. It is worthwhile putting some effort into optimizing the efficiency of the arrangement, and in principle it is possible to handle a very large number of classes based on distance (Kräutler and Hünenberger, 2006).

5.5 When the dust has settled

At the end of the central loop of the program, a new configuration of molecules is created, and there are a number of important configurational properties that can be calculated. At this point in the program, the potential energy and the forces on particular molecules are available. These quantities, and functions derived from them, may be added to the accumulators used to eventually calculate simulation run averages. For example, the square of the configurational energy, $\mathcal{V}^2$, is calculated so that, at the end of the simulation, $\langle \mathcal{V}^2 \rangle - \langle \mathcal{V} \rangle^2$ can be used in eqns (2.80) or (2.89) to calculate the specific heat in the canonical or microcanonical ensemble. Although the average force and torque on a molecule in a fluid are zero, the mean-square values of these properties can be used to calculate the quantum corrections to the free energy given by eqns (2.153), (2.155). In an MD simulation, the force $\mathbf{f}_i$ on molecule i from its neighbours is calculated anyway, to

move the molecules. The forces are not required in the implementation of the Metropolis MC method, so that they must be evaluated in addition to the potential energy if the quantum corrections are to be evaluated. The alternative of calculating the corrections via $g(r)$ (eqn (2.154)) is less accurate. The mean-square forces, as well as the average of the Laplacian of the potential, are needed to calculate the configurational temperature of eqn (2.56) and Appendix F.

This is the point in the simulation at which a direct calculation of the chemical potential can be carried out. A test particle, which is identical to the other molecules in the simulation, is inserted into the fluid at random (Widom, 1963). The particle does not disturb the phase trajectory of the fluid, but the energy of interaction with the other molecules, $\mathcal{V}_{\text{test}}$, is calculated. This operation is repeated many times, and the quantity $\exp(-\beta\mathcal{V}_{\text{test}})$ is used in eqn (2.75) to compute the chemical potential. In the MD method, the total kinetic temperature, $\mathcal{T}$, fluctuates, and it is essential to use eqn (2.76a). The insertion subroutine increases the running time somewhat; a figure of 20 % is typical.

The difficulty with this method is that a large number of substantial overlaps occur when particles are inserted. The exponential is then negligible, and we do not improve the statistics in the estimation of μ. Special techniques may be needed in such cases and we address these in Chapter 9. This is not a severe problem for the Lennard-Jones fluid, where Powles et al. (1982) have calculated μ close to the triple point with runs of less than 8000 timesteps. For molecular fluids, Romano and Singer (1979) calculated μ for a model of liquid chlorine up to a reduced density of $\rho\sigma^3 = 0.4$ (the triple point density is ≈ 0.52); Fincham et al. (1986) obtained an accurate estimate of μ by direct insertion at $\rho\sigma^3 = 0.45$ for the same model. In the case of mixtures, the chemical potential of each species can be determined by separately inserting molecules of that species.

The calculation of μ for a chain molecule, such as a small polymer in a polymer melt or solution, is more difficult. Once the first monomer of the ghost chain has been randomly inserted, without a significant overlap, the addition of subsequent monomers is likely to lead to such overlaps and, even for chains of modest length, the overall trial insertion will result in a zero contribution to μ. The configurational-bias method (see Section 9.3.4) can be used to thread the inserted chain through the fluid so that it makes a reasonable contribution to μ (Willemsen et al., 1998).

In calculating μ for an ionic fluid, charge neutrality can be preserved by inserting a pair of oppositely charged test particles and calculating the Boltzmann factor of the energy of the test ion pair with the ionic fluid. The two charged particles can be inserted at random positions in the fluid, but the accuracy of the calculation is improved if we use a Rosenbluth approach to increase the likelihood that the inserted pairs are separated by a short distance, that is, close to contact in the primitive model of the electrolyte (Orkoulas and Panagiotopoulos, 1993). More accurate calculations of the chemical potential of ions in aqueous environments can be made using methods such as MBAR (Mester and Panagiotopoulos, 2015); for further discussion see Section 9.2.4.

For systems of hard molecules, the pressure is generally not calculated in the conventional way: the potential energy is not differentiable, and so it is impossible to define forces. In MD of such systems, as described in Section 3.7, the program proceeds from collision to collision. At each event, the collisional virial may be calculated, from the impulse (rather than force) acting between the particles. This is accumulated in a time average and, at

intervals, these averages are used to calculate the excess part of the pressure, as described in Section 2.4. However, for nonspherical hard particle models (Section 3.7.2), event-driven MD is a more complicated proposition than MC.

In MC simulations of hard-particle systems, the pressure may be calculated numerically by a box-scaling procedure, first introduced by Eppenga and Frenkel (1984). The problem has been revisited several times (Allen, 2006c; de Miguel and Jackson, 2006; Brumby et al., 2011; Jiménez-Serratos et al., 2012). Start with the canonical ensemble expression

$$\begin{aligned}\frac{P^{\mathrm{ex}}}{k_{\mathrm{B}}T} &= -\frac{1}{k_{\mathrm{B}}T}\frac{\partial A^{\mathrm{ex}}}{\partial V} = \frac{\partial \ln Q^{\mathrm{ex}}}{\partial V} = \frac{1}{Q^{\mathrm{ex}}}\frac{\partial Q^{\mathrm{ex}}}{\partial V} \\ &= \lim_{\Delta V\to 0_+}\frac{1}{\Delta V}\frac{Q^{\mathrm{ex}}(V)-Q^{\mathrm{ex}}(V-\Delta V)}{Q^{\mathrm{ex}}(V)} \\ &= \lim_{\Delta V\to 0_+}\frac{1}{\Delta V}\left(1-\frac{Q^{\mathrm{ex}}(V-\Delta V)}{Q^{\mathrm{ex}}(V)}\right).\end{aligned}$$

Here we are considering a system that has been reduced in volume by a small amount ΔV relative to the system of interest. We can relate the ratio of partition functions to a single quantity, averaged over particle coordinates, if we take the latter to be scaled homogeneously, along with the box lengths

$$L' = fL, \qquad \mathbf{r}' = f\mathbf{r}, \qquad \text{where} \quad f = \left(\frac{V-\Delta V}{V}\right)^{1/3}.$$

The ratio $Q^{\mathrm{ex}}(V-\Delta V)/Q^{\mathrm{ex}}(V)$ may be interpreted as the relative statistical weights of these systems. For hard particles, the statistical weights of configurations are simply zero (if there is an overlap) or one (if there is no overlap): therefore this ratio is the average probability of *no overlap* being generated by the volume scaling. Eppenga and Frenkel (1984) argue that, for sufficiently small ΔV, this may be written as a product of independent probabilities of no overlap for all the pairs, each of which can be written as $1-p$ where p is the probability that the volume scaling results in an overlap of a given pair; this value is the same for all pairs. Hence

$$\frac{P^{\mathrm{ex}}}{k_{\mathrm{B}}T} = \lim_{\Delta V\to 0_+}\frac{1-\prod_{i<j}(1-p)}{\Delta V} = \lim_{\Delta V\to 0_+}\frac{\sum_{i<j}p}{\Delta V} = \lim_{\Delta V\to 0_+}\frac{\left\langle N_{\mathrm{overlap}}\right\rangle}{\Delta V},$$

where we use the fact that $p \ll 1$ for small ΔV. Therefore we may calculate P^{ex} by MC simulations in which, at intervals, we count the overlaps N_{overlap} (summed over all distinct pairs) that would result from an isotropic scaling of the box and particle coordinates by the factor f just defined, average them and divide by ΔV. Alternatively, N_{overlap} can be calculated by counting the overlaps that would result from scaling the particle dimensions up, isotropically, by a factor $1/f$, keeping the coordinates fixed. In practice, an extrapolation to low ΔV may be needed.

As just described, the method is restricted to *convex* hard molecules: an expansion (rather than a contraction) of the box cannot produce any overlaps in this case. For non-convex particles (for example, hard dumb-bells) an expansion can produce overlaps and the prescription must be modified: Brumby et al. (2011) discuss how to take this into

account, as well as some of the subtleties of the statistical mechanics. Jiménez-Serratos et al. (2012) explain how to handle the case of finite discontinuities in the potential, as seen for square wells, for instance. It is possible to extend the procedure to calculate all the components of the pressure tensor, either by using the geometry of convex particles (Allen, 2006c) or by scaling separately in the x, y, and z directions (de Miguel and Jackson, 2006; Brumby et al., 2011). There are several ways of deriving the expression for P^{ex} (de Miguel and Jackson, 2006), and it is also possible to relate it to the pair distribution function at contact (Boublik, 1974), which is a well-known statistical mechanical route to the pressure. The use of this approach to estimate the surface tension of interfaces (Gloor et al., 2005) will be discussed in Chapter 14.

In this book, we have assumed that calculation of most other properties of interest will be carried out after the simulation, by analysis of output configurations stored on disk or other media. This analysis will be the subject of Chapter 8. In some cases, however, it may be preferable to calculate properties such as time correlation functions and structural distributions during the simulation itself. Against this, it must be said that the simulation program may be slowed down unnecessarily if too much calculation is included in it. For example, the pair distribution function $g(r)$ involves a sum over pairs of molecules, and this is sometimes included in the inner loop of an MD program. However, $g(r)$ is generally of interest for separations r much greater than the potential cutoff r_c, and so we need to examine many more pairs than would be required to calculate the energy and forces. Also, successive configurations are likely to be highly correlated. It is sensible to carry out this expensive summation less frequently than once per timestep. Similar observations apply to many other properties. The cleanest approach is to write a completely separate routine for calculating $g(r)$, and call it (say) every 10–20 steps or MC cycles; or alternatively to perform the analysis afterwards from stored configurations. In a similar way, time correlation functions may be calculated during an MD simulation, by methods very similar to some of those described in Chapter 8. However, this will require extra storage in the simulation program, and will make the program itself more complicated.

5.6 Starting up

In the remainder of this chapter, we consider the overall structure of the simulation programs. It is quite common to carry out sequences of runs at different state points, each run following on from the previous one. By this means, the starting configuration for most runs is obtained from a nearby state point, and will not require as long to equilibrate as one prepared from scratch. On the other hand, with the availability of parallel computing resources that can be used as 'task farms', the overall workflow may be more efficient if many runs can be conducted simultaneously. In this case, each one will require an independent starting point.

In both MD and MC techniques, therefore, it is necessary to design a starting configuration for the first simulation of a sequence. For MC, the molecular positions and orientations are specified, and for MD, in addition, the initial velocities and angular velocities must be chosen. Assuming that the liquid state is the target, it is important to choose a configuration that can relax quickly to the structure and velocity distribution appropriate to a fluid. This period of equilibration must be monitored carefully, since the disappearance of the initial structure may be quite slow. As a series of runs progresses, the coordinates

and velocities from the last configuration of the previous run can be scaled (giving a new density, energy, etc.) to form the initial configuration for the next run. Again, with each change in state point, a period of equilibration must be set aside before attempting to compute proper simulation averages.

5.6.1 The initial configuration

The simplest method of constructing a liquid structure is to place molecules at random inside the simulation box (see Appendix E). The difficulty with this technique is that the configuration so constructed may contain substantial overlaps. This would be totally unphysical for a hard-core system, for which the potential energy would be infinite. For soft potentials, the energy for most random configurations, although high, can be calculated (provided no two molecules are centred at exactly the same point), so this type of configuration can be used to start Monte Carlo simulations, provided that the system is allowed to relax. In molecular dynamics, on the other hand, the large intermolecular potentials and the correspondingly large forces can cause difficulties in the solution of the differential equations of motion. A standard approach is to begin with a period of energy minimization: effectively, the velocities are set to zero at the start of each step, so the system evolves in the direction of the forces ('downhill'). During this period, a maximum limit can be set on the individual atom displacements, to avoid the creation of further overlaps. Another strategy is to replace the real potential by a version in which the repulsive core is reduced in size and/or strength, and slowly restore the full potential during equilibration.

It is more usual to start from a lattice. Almost any lattice is suitable, but historically the face-centred cubic structure, with its $4M^3$ (M = integer) lattice points has been the starting configuration for many simulations. This lattice is shown in Fig. 5.6. The lattice spacing is chosen so that the appropriate liquid state density is obtained. During the course of the simulation the lattice structure will disappear, to be replaced by a typical liquid structure. This process of 'melting' can be enhanced by giving each molecule a small random displacement from its initial lattice point along each of the space-fixed axes (Schofield, 1973).

In the case of a molecular fluid, it is also necessary to assign the initial orientations of the molecules. A model commonly used for linear molecules is the α-FCC lattice, which is the solid structure of CO_2 and one of the phases of N_2 (see Fig. 5.6). In this structure, there are four sublattices of molecules oriented along the four diagonals of the unit cell. A code for generating the α-FCC lattice is given in Code 5.6. For non-linear molecules, any suitable known crystal structure could be used. Small random displacements can also be applied to the lattice orientations so as to speed up melting. Some workers prefer to choose the orientations completely randomly given a centre of mass structure, although at high densities, with elongated molecules, random assignment of the directions can result in non-physical overlaps.

5.6.2 The initial velocities

For a molecular dynamics simulation, the initial velocities of all the molecules must be specified. It is usual to choose random velocities, with magnitudes conforming to the