CHAPTER SEVEN

Molecular Dynamics Simulation Methods

7.1 Introduction

In molecular dynamics, successive configurations of the system are generated by integrating Newton’s laws of motion. The result is a trajectory that specifies how the positions and velocities of the particles in the system vary with time. Newton’s laws of motion can be stated as follows:

1. A body continues to move in a straight line at constant velocity unless a force acts upon it.
2. Force equals the rate of change of momentum.
3. To every action there is an equal and opposite reaction.

The trajectory is obtained by solving the differential equations embodied in Newton’s second law \( F = ma \):

\[
\frac{d^2 x_i}{dt^2} = \frac{F_{x_i}}{m_i}
\]  

(7.1)

This equation describes the motion of a particle of mass \( m_i \) along one coordinate \( x_i \) with \( F_{x_i} \) being the force on the particle in that direction.

It is helpful to distinguish three different types of problem to which Newton’s laws of motion may be applied. In the simplest case, no force acts on each particle between collisions. From one collision to the next, the position of the particle thus changes by \( v_i \delta t \), where \( v_i \) is the (constant) velocity and \( \delta t \) is the time between collisions. In the second situation, the particle experiences a constant force between collisions. An example of this type of motion would be that of a charged particle moving in a uniform electric field. In the third case, the force on the particle depends on its position relative to the other particles. Here the motion is often very difficult, if not impossible, to describe analytically, due to the coupled nature of the particles’ motions.

7.2 Molecular Dynamics Using Simple Models

The first molecular dynamics simulation of a condensed phase system was performed by Alder and Wainwright in 1957 using a hard-sphere model [Alder and Wainwright 1957]. In this model, the spheres move at constant velocity in straight lines between collisions. All collisions are perfectly elastic and occur when the separation between the centres of
the spheres equals the sphere diameter. The pair potential thus has the form shown in Figure 7.1. Some early simulations also used the square-well potential, where the interaction energy between two particles is zero beyond a cutoff distance $\sigma_2$; infinite below a smaller cutoff distance $\sigma_1$; and equal to $V_0$ between the two cutoff values (Figure 7.1). The steps involved in the hard-sphere calculation are as follows:

1. Identify the next pair of spheres to collide and calculate when the collision will occur.
2. Calculate the positions of all the spheres at the collision time.
3. Determine the new velocities of the two colliding spheres after the collision.
4. Repeat from 1 until finished.

The new velocities of the colliding spheres are calculated by applying the principle of conservation of linear momentum.

Simple interaction models such as the hard-sphere potential obviously suffer from many deficiencies but have nevertheless provided many useful insights into the microscopic

---

Fig. 7.1: The hard-sphere and square-well potentials.

Fig. 7.2: Molecular graphics representation of the paths generated by 32 hard spherical particles in the solid (left) and fluid (right) phase. (Reproduced from Alder B J and T E Weinwright 1959, Studies in Molecular Dynamics. I. General Method. Journal of Chemical Physics 31: 459-466.)
nature of fluids. The early workers were particularly keen to quantify the differences between the solid and fluid phases; it is interesting to note that such investigations were facilitated by early molecular graphics systems, which enabled the trajectories of the particles to be represented simultaneously (Figure 7.2).

7.3 Molecular Dynamics with Continuous Potentials

In more realistic models of intermolecular interactions, the force on each particle will change whenever the particle changes its position, or whenever any of the other particles with which it interacts changes position. The first simulation using continuous potentials was of argon by Rahman [Rahman 1964], who also performed the first simulation of a molecular liquid (water) [Rahman and Stillinger 1971] and made many other important methodological contributions in molecular dynamics. Under the influence of a continuous potential the motions of all the particles are coupled together, giving rise to a many-body problem that cannot be solved analytically. Under such circumstances the equations of motion are integrated using a finite difference method.

7.3.1 Finite Difference Methods

Finite difference techniques are used to generate molecular dynamics trajectories with continuous potential models, which we will assume to be pairwise additive. The essential idea is that the integration is broken down into many small stages, each separated in time by a fixed time $\delta t$. The total force on each particle in the configuration at a time $t$ is calculated as the vector sum of its interactions with other particles. From the force we can determine the accelerations of the particles, which are then combined with the positions and velocities at a time $t$ to calculate the positions and velocities at a time $t+\delta t$. The force is assumed to be constant during the time step. The forces on the particles in their new positions are then determined, leading to new positions and velocities at time $t+2\delta t$, and so on.

There are many algorithms for integrating the equations of motion using finite difference methods, several of which are commonly used in molecular dynamics calculations. All algorithms assume that the positions and dynamic properties (velocities, accelerations, etc.) can be approximated as Taylor series expansions:

$$r(t+\delta t) = r(t) + \delta tv(t) + \frac{1}{2} \delta^2 a(t) + \frac{1}{6} \delta^3 b(t) + \frac{1}{24} \delta^4 c(t) + \cdots \quad (7.2)$$

$$v(t+\delta t) = v(t) + \delta ta(t) + \frac{1}{2} \delta^2 b(t) + \frac{1}{6} \delta^3 c(t) + \cdots \quad (7.3)$$

$$a(t+\delta t) = a(t) + \delta tb(t) + \frac{1}{2} \delta^2 c(t) + \cdots \quad (7.4)$$

$$b(t+\delta t) = b(t) + \delta tc(t) + \cdots \quad (7.5)$$

where $v$ is the velocity (the first derivative of the positions with respect to time), $a$ is the acceleration (the second derivative), $b$ is the third derivative, and so on. The Verlet algorithm [Verlet 1967] is probably the most widely used method for integrating the equations of
motion in a molecular dynamics simulation. The Verlet algorithm uses the positions and accelerations at time \( t \), and the positions from the previous step, \( r(t - \delta t) \), to calculate the new positions at \( t + \delta t \), \( r(t + \delta t) \). We can write down the following relationships between these quantities and the velocities at time \( t \):

\[
\begin{align*}
\mathbf{r}(t + \delta t) &= \mathbf{r}(t) + \delta t \mathbf{v}(t) + \frac{1}{2} \delta^2 \mathbf{a}(t) + \cdots \\
\mathbf{r}(t - \delta t) &= \mathbf{r}(t) - \delta t \mathbf{v}(t) + \frac{1}{2} \delta^2 \mathbf{a}(t) - \cdots
\end{align*}
\] (7.6) (7.7)

Adding these two equations gives

\[
\mathbf{r}(t + \delta t) = 2\mathbf{r}(t) - \mathbf{r}(t - \delta t) + \delta^2 \mathbf{a}(t)
\] (7.8)

The velocities do not explicitly appear in the Verlet integration algorithm. The velocities can be calculated in a variety of ways; a simple approach is to divide the difference in positions at times \( t + \delta t \) and \( t - \delta t \) by \( 2\delta t \):

\[
\mathbf{v}(t) = [\mathbf{r}(t + \delta t) - \mathbf{r}(t - \delta t)] / 2\delta t
\] (7.9)

Alternatively, the velocities can be estimated at the half-step, \( t + \frac{1}{2} \delta t \):

\[
\mathbf{v}(t + \frac{1}{2} \delta t) = [\mathbf{r}(t + \delta t) - \mathbf{r}(t)] / \delta t
\] (7.10)

Implementation of the Verlet algorithm is straightforward and the storage requirements are modest, comprising two sets of positions (\( r(t) \) and \( r(t - \delta t) \)) and the accelerations \( \mathbf{a}(t) \). One of its drawbacks is that the positions \( r(t + \delta t) \) are obtained by adding a small term (\( \delta^2 \mathbf{a}(t) \)) to the difference of two much larger terms, \( 2\mathbf{r}(t) \) and \( \mathbf{r}(t - \delta t) \). This may lead to a loss of precision. The Verlet algorithm has some other disadvantages. The lack of an explicit velocity term in the equations makes it difficult to obtain the velocities, and indeed the velocities are not available until the positions have been computed at the next step. In addition, it is not a self-starting algorithm; the new positions are obtained from the current positions \( r(t) \) and the positions from the previous time step, \( r(t - \delta t) \). At \( t = 0 \) there is obviously only one set of positions and so it is necessary to employ some other means to obtain positions at \( t - \delta t \). One way to obtain \( r(t - \delta t) \) is to use the Taylor series, Equation (7.2), truncated after the first term. Thus, \( r(t - \delta t) = r(0) - \delta t \mathbf{v}(0) - \delta^2 \mathbf{a}(0) \).

Several variations on the Verlet algorithm have been developed. The leap-frog algorithm [Hockney 1970] uses the following relationships:

\[
\begin{align*}
\mathbf{r}(t + \delta t) &= \mathbf{r}(t) + \delta t \mathbf{v}(t + \frac{1}{2} \delta t) \\
\mathbf{v}(t + \frac{1}{2} \delta t) &= \mathbf{v}(t - \frac{1}{2} \delta t) + \delta t \mathbf{a}(t)
\end{align*}
\] (7.11) (7.12)

To implement the leap-frog algorithm, the velocities \( \mathbf{v}(t + \frac{1}{2} \delta t) \) are first calculated from the velocities at time \( t - \frac{1}{2} \delta t \) and the accelerations at time \( t \). The positions \( r(t + \delta t) \) are then deduced from the velocities just calculated together with the positions at time \( r(t) \) using Equation (7.11). The velocities at time \( t \) can be calculated from

\[
\mathbf{v}(t) = \frac{1}{2} [\mathbf{v}(t + \frac{1}{2} \delta t) + \mathbf{v}(t - \frac{1}{2} \delta t)]
\] (7.13)

The velocities thus ‘leap-frog’ over the positions to give their values at \( t + \frac{1}{2} \delta t \) (hence the name). The positions then leap over the velocities to give their new values at \( t + \delta t \), ready for the velocities at \( t + \frac{3}{2} \delta t \), and so on. The leap-frog method has two advantages over the
standard Verlet algorithm: it explicitly includes the velocity and also does not require the calculation of the differences of large numbers. However, it has the obvious disadvantage that the positions and velocities are not synchronised. This means that it is not possible to calculate the kinetic energy contribution to the total energy at the same time as the positions are defined (from which the potential energy is determined).

The velocity Verlet method [Swope et al. 1982] gives positions, velocities and accelerations at the same time and does not compromise precision:

\[ r(t + \delta t) = r(t) + \delta t v(t) + \frac{1}{2} \delta t^2 a(t) \]  
\[ v(t + \delta t) = v(t) + \frac{1}{2} \delta t [a(t) + a(t + \delta t)] \]  

(7.14)  
(7.15)

The velocity Verlet method is actually implemented as a three-stage procedure because, as can be seen from Equation (7.15), to calculate the new velocities requires the accelerations at both \( t \) and \( t + \delta t \). Thus in the first step the positions at \( t + \delta t \) are calculated according to Equation (7.14) using the velocities and the accelerations at time \( t \). The velocities at time \( t + \frac{1}{2} \delta t \) are then determined using:

\[ v(t + \frac{1}{2} \delta t) = v(t) + \frac{1}{2} \delta t a(t) \]  

(7.16)

New forces are next computed from the current positions, thus giving \( a(t + \delta t) \). In the final step, the velocities at time \( t + \delta t \) are determined using:

\[ v(t + \delta t) = v(t + \frac{1}{2} \delta t) + \frac{1}{2} \delta t a(t + \delta t) \]  

(7.17)

*Beeman's algorithm [Beeman 1976]* is also related to the Verlet method:

\[ r(t + \delta t) = r(t) + \delta t v(t) + \frac{3}{2} \delta t^2 a(t) - \frac{1}{2} \delta t^2 a(t - \delta t) \]  
\[ v(t + \delta t) = v(t) + \frac{3}{2} \delta t a(t + \delta t) + \frac{3}{2} \delta t a(t) - \frac{1}{2} \delta t a(t - \delta t) \]  

(7.18)  
(7.19)

The Beeman integration scheme uses a more accurate expression for the velocity. As a consequence it often gives better energy conservation, because the kinetic energy is calculated directly from the velocities. However, the expressions used are more complex than those of the Verlet algorithm and so it is computationally more expensive.

We have already encountered four different integration methods, with more to come! Why should we use one method in preference to another? What features characterise a ‘good’ integration method? As with any other computer algorithm, an ideal integration scheme should be fast, require minimal memory and be easy to program. However, for most molecular dynamics simulations these issues are of secondary importance; most calculations do not make significant memory demands of even a modest workstation, and the time required for the integration is usually trivial compared to the other parts of the calculation. The most demanding part of a molecular dynamics simulation is invariably the calculation of the force on each particle in the system. More important considerations are that the integration algorithm should conserve energy and momentum, be time-reversible, and should permit a long time step, \( \delta t \), to be used. The size of the time step is particularly relevant to the computational demands as a simulation using a long time step will require fewer iterations to cover a given amount of phase space. A less important requirement is that the integration algorithm should give the same results as an exact, analytical trajectory...
(this can be tested using simple problems for which an analytical solution can be derived). We would, in any case, expect the calculated trajectory to deviate from the exact trajectory because the computer can only store numbers to a given precision.

The order of an integration method is the degree to which the Taylor series expansion, Equation (7.2), is truncated: it is the lowest term that is not present in the expansion. The order may not always be apparent from the formulae used. For example, the highest-order derivative that appears in the Verlet formulae is the second, \( a(t) \), yet the Verlet algorithm is, in fact, a fourth-order method. This is because the third-order terms, which cancel when Equation (7.6) is added to Equation (7.7), are still implied in the expansion:

\[
\begin{align*}
\mathbf{r}(t + \delta t) &= \mathbf{r}(t) + \delta \mathbf{v}(t) + \frac{1}{2} \delta^2 \mathbf{a}(t) + \frac{1}{2} \delta t \mathbf{b}(t) + \frac{1}{3} \delta^3 \mathbf{c}(t) \\
\mathbf{r}(t - \delta t) &= \mathbf{r}(t) - \delta \mathbf{v}(t) + \frac{1}{2} \delta^2 \mathbf{a}(t) - \frac{1}{2} \delta t \mathbf{b}(t) + \frac{1}{3} \delta^3 \mathbf{c}(t)
\end{align*}
\]

(7.20)

(7.21)

### 7.3.2 Predictor–Corrector Integration Methods

The predictor-corrector methods [Gear 1971] form a general family of integration algorithms from which one can select a scheme that is correct to a given order. These methods have three basic steps. First, new positions, velocities, accelerations and higher-order terms are predicted according to the Taylor expansion, Equations (7.2)–(7.4). In the second stage, the forces are evaluated at the new positions to give accelerations \( a(t + \delta t) \). These accelerations are then compared with the accelerations that are predicted from the Taylor series expansion, \( \mathbf{a}^*(t + \delta t) \). The difference between the predicted and calculated accelerations is then used to 'correct' the positions, velocities, etc., in the correction step:

\[
\Delta \mathbf{a}(t + \delta t) = \mathbf{a}^*(t + \delta t) - \mathbf{a}(t + \delta t)
\]

(7.22)

Then

\[
\begin{align*}
\mathbf{r}^*(t + \delta t) &= \mathbf{r}(t) + \delta \mathbf{v}(t) + c_0 \Delta \mathbf{a}(t + \delta t) \\
\mathbf{v}^*(t + \delta t) &= \mathbf{v}(t) + c_1 \Delta \mathbf{a}(t + \delta t) \\
\mathbf{a}^*(t + \delta t)/2 &= \mathbf{a}(t + \delta t)/2 + c_2 \Delta \mathbf{a}(t + \delta t) \\
\mathbf{b}^*(t + \delta t)/6 &= \mathbf{b}(t + \delta t)/6 + c_3 \Delta \mathbf{a}(t + \delta t)
\end{align*}
\]

(7.23)

(7.24)

(7.25)

(7.26)

Gear has suggested 'best' values of the coefficients \( c_0, c_1, \ldots \). The set of coefficients to use depends upon the order of the Taylor series expansion. In Equations (7.23)–(7.26) the expansion has been truncated after the third derivative of the positions (i.e. \( \mathbf{b}(t) \)). The appropriate set of coefficients to use in this case is \( c_0 = \frac{1}{6}, c_1 = \frac{1}{6}, c_2 = 1 \) and \( c_3 = \frac{1}{3} \).

The storage required for the Gear predictor-corrector algorithm is \( 3 \times (O + 1)N \), where \( O \) is the highest-order differential used in the Taylor series expansion and \( N \) is the number of atoms. Thus the storage required for our example is \( 15N \), which is rather more than for the Verlet algorithm, which uses \( 9N \). More importantly, the Gear algorithm requires two time-consuming force evaluations per time step, though this is not necessarily a disadvantage as it may permit a time step more than twice as long as an alternative algorithm.
There are many variants of the `predictor-corrector` theme; of these, we will only mention the algorithm used by Rahman in the first molecular dynamics simulations with continuous potentials [Rahman 1964]. In this method, the first step is to predict new positions as follows:

\[
\mathbf{r}(t + \delta t) = \mathbf{r}(t - \delta t) + 2\delta t\mathbf{v}(t) \tag{7.27}
\]

New accelerations are calculated at these new positions in the usual way. These accelerations are then used to generate a set of new velocities, and then corrected positions:

\[
\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \frac{1}{2}\delta t(a(t + \delta t) + a(t)) \tag{7.28}
\]

\[
\mathbf{r}^\prime(t + \delta t) = \mathbf{r}(t) + \frac{1}{2}\delta t(\mathbf{v}(t) + \mathbf{v}(t + \delta t)) \tag{7.29}
\]

The acceleration can then be recalculated at the new corrected positions to give new velocities. The method then iterates over the two Equations (7.28) and (7.29). Two or three passes are usually required to achieve consistency, with a force evaluation at each step. The computational demands of this scheme mean that it is now rarely used, though it does give accurate solutions of the equations of motion.

### 7.3.3 Which Integration Algorithm is Most Appropriate?

The wide variety of integration schemes available can make it difficult to decide which is the most appropriate one to use. Various factors may need to be taken into account when deciding which is most appropriate. Clearly, the computational effort required is a major consideration. As we have already indicated, an algorithm that is nominally more expensive (for example, because it requires more than one force evaluation per iteration) may permit a significantly longer time step to be used and so, in fact, be more cost-effective. One of the most important considerations is energy conservation; this can be calculated as the root-mean-square fluctuation and is often plotted against the time step, as shown in Figure 7.3. In Appendix 7.1 we show why energy conservation would be expected in a molecular dynamics simulation. The kinetic and potential energy components would be expected to fluctuate in equal and opposite directions; this is also shown in Figure 7.3.

As the time step increases, so the RMS energy fluctuation also increases. For the argon simulation reported in Figure 7.3, the RMS fluctuation in the total energy is approximately 0.006 kcal/mol and the RMS fluctuations in the kinetic and potential energies are approximately 2.5 kcal/mol. With a time step of 25 fs the RMS fluctuation rises to 0.04 kcal/mol and with a time step of 5 fs the value is 0.002 kcal/mol. A variation of one part in \(10^6\) is generally considered acceptable. The different algorithms may vary in the rate at which the error varies with the time step. For example, it has been shown that for short time steps the predictor-corrector methods may be more accurate, but for longer time steps the Verlet algorithm may be better [Fincham and Heyes 1982]. Other factors that may be important when choosing an integration algorithm include the memory required; the synchronisation of positions and velocities; whether they are self-starting (some methods require properties at \(t - \delta t\), which obviously do not exist); and
whether it is possible to perform simulations in other ensembles such as the isothermal-isobaric ensemble.

7.3.4 Choosing the Time Step

There are no hard and fast rules for calculating the most appropriate time step to use in a molecular dynamics simulation; too small and the trajectory will cover only a limited proportion of the phase space; too large and instabilities may arise in the integration algorithm due to high energy overlaps between atoms. These two extremes are illustrated in Figure 7.4. Such instabilities would certainly lead to a violation of energy and linear momentum conservation and could result in a program failure due to numerical overflow. The effect of changing the time step on the dynamics can be illustrated using a very simple system
Consisting of two argon atoms interacting under the Lennard-Jones potential. The behaviour of this system can be determined analytically and so compared with the numerical integration. Suppose the argon atoms are moving towards each other along the $x$ axis with initial velocities of $353 \text{ m/s}$ (this corresponds to the most probable speed of argon at 300 K). We can then plot how the interatomic distance varies with time and compare it to the analytical potential. The result obtained using two time steps (10 fs and 50 fs) are shown in Figure 7.5. In both cases the numerical trajectory initially lags behind the analytical one, but then as the atoms pass through their minimum energy separation and move up the repulsive barrier the atoms ‘jump through’ the energy barrier. This leads to a gain in energy and the atoms then move apart with velocities that are slightly too high. In both numerical trajectories the total energy rises after the collision. Unfortunately, the atoms move most quickly and take the largest steps in the very region (i.e. near the energy minimum) where it would be best to take the smallest steps. The total error is correlated with the time step, with the largest errors arising for the largest time steps. Of course, with a small time step much more computer time will be required for a given length of calculation; the aim is to find the correct balance between simulating the ‘correct’ trajectory and covering the phase space. If the time

**Fig. 7.5**: Difference between the exact and numerical trajectories for the approach of two argon atoms with time steps of 10 fs and 50 fs.
step is too large, then the trajectory will 'blow up', as can be seen for the argon dimer system with a time step of 100 fs (Figure 7.6).

When simulating an atomic fluid the time step should be small compared to the mean time between collisions. When simulating flexible molecules a useful guide is that the time step should be approximately one-tenth the time of the shortest period of motion. In flexible molecules, the highest-frequency vibrations are due to bond stretches, especially those of bonds to hydrogen atoms. A C–H bond vibrates with a repeat period of approximately 10 fs. The timescales of some typical motions together with appropriate time steps are shown in Table 7.1, which can be used to choose an appropriate time step.

The requirement that the time step is approximately one order of magnitude smaller than the shortest motion is clearly a severe restriction, particularly as these high-frequency motions are usually of relatively little interest and have a minimal effect on the overall behaviour of the system. One solution to this problem is to 'freeze out' such vibrations by constraining the appropriate bonds to their equilibrium values while still permitting the rest of the degrees of freedom to vary under the intramolecular and intermolecular forces present. This enables a longer time step to be used. We will consider such constraint dynamics methods in Section 7.5.

<table>
<thead>
<tr>
<th>System</th>
<th>Types of motion present</th>
<th>Suggested time step (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atoms</td>
<td>Translation</td>
<td>$10^{-14}$</td>
</tr>
<tr>
<td>Rigid molecules</td>
<td>Translation and rotation</td>
<td>$5 \times 10^{-15}$</td>
</tr>
<tr>
<td>Flexible molecules, rigid bonds</td>
<td>Translation, rotation, torsion</td>
<td>$2 \times 10^{-15}$</td>
</tr>
<tr>
<td>Flexible molecules, flexible bonds</td>
<td>Translation, rotation, torsion, vibration</td>
<td>$10^{-15}$ or $5 \times 10^{-16}$</td>
</tr>
</tbody>
</table>

Table 7.1 The different types of motion present in various systems together with suggested time steps.
7.3.5 Multiple Time Step Dynamics

Table 7.1 presents us with something of a dilemma. We would obviously desire to explore as much of the phase space as possible but this may be compromised by the need for a small time step. One possible approach is to use a multiple time step method. The underlying rationale is that certain interactions evolve more rapidly with time than other interactions. The twin-range method (Section 6.7.1) is a crude type of multiple time step approach, in that interactions involving atoms between the lower and upper cutoff distance remain constant and change only when the neighbour list is updated. However, this approach can lead to an accumulation of numerical errors in calculated properties. A more sophisticated approach is to approximate the forces due to these atoms using a Taylor series expansion [Streett et al. 1978]:

\[ f(t + \tau \delta t) = f(t) + (\tau \delta t) \frac{df(t)}{dt} + \frac{1}{2} (\tau \delta t)^2 \frac{d^2f(t)}{dt^2} + \cdots \]  

(7.30)

This series expansion is truncated at a specified order and is probably most easily implemented within a predictor-corrector type of algorithm, where the higher-order terms are already computed. This method has been applied to relatively simple systems such as molecular fluids [Streett et al. 1978] and alkane chain liquids [Swindoll and Halle 1984].

An alternative formulation of a multiple time step method is the 'reversible reference system propagation algorithm' (r-RESPA) method [Tuckerman et al. 1992]. In this method, the forces within a system are classified into a number of groups according to how rapidly the force varies over time. Each group then has its own time step while maintaining accuracy and numerical stability. The starting point for this algorithm is the Liouville equation, which defines how the state of the system, \( \Gamma(t) \), evolves over time:

\[ \Gamma(t) = \mathcal{U} \Gamma(t = 0) \]  

(7.31)

The exponential \( \exp(itA) \) in Equation (7.31) involves the so-called Liouville operator, \( \mathcal{L} \), which in the case of a molecular system containing \( N \) atoms (and so \( 3N \) coordinates) can be expressed:

\[ i\mathcal{L} = \sum_{i=1}^{3N} \left[ \frac{\partial}{\partial \xi_i} \frac{\partial}{\partial \xi_i} + F_i(x) \frac{\partial}{\partial p_i} \right] \]  

(7.32)

In the r-RESPA method this operator is decomposed into two or more parts, for example:

\[ \mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3 + \mathcal{L}_4 \]  

(7.33)

Each of these parts is then associated with specific terms in the force equation. For example, \( \mathcal{L}_1 \) may correspond to the bond-stretching terms, \( \mathcal{L}_2 \) to the angle-bending and torsional terms, \( \mathcal{L}_3 \) to the short-range non-bonded interactions and \( \mathcal{L}_4 \) to the long-range non-bonded interactions. Suppose the time step with which we evaluate the bond-stretching terms is \( \delta t_1 \). Integers \( n_1, n_2 \) and \( n_3 \) then define the time steps for the three other forces as follows:

\[ \delta t_2 = n_1 \delta t_1; \quad \delta t_3 = n_2 \delta t_1; \quad \delta t_4 = n_1 n_2 n_3 \delta t_1 \]  

(7.34)

The underlying theory of r-RESPA is somewhat involved, but the final result and consequent implementation is actually rather straightforward, being very closely related to the velocity Verlet integration scheme. For our four-way decomposition the algorithm would
be implemented as follows:

Calculate forces-1 (i.e. \(a_1(t)\))
Calculate forces-2 (i.e. \(a_2(t)\))
Calculate forces-3 (i.e. \(a_3(t)\))
Calculate forces-4 (i.e. \(a_4(t)\))

\[
\text{do} \quad \text{step} = 1, N_{\text{steps}}
\]

\[
v = v + \frac{1}{2} n_1 n_2 n_3 \delta t \dot{a}_4 \\
\text{do} \quad i_3 = 1, n_3
\]

\[
v = v + \frac{1}{2} n_1 n_2 \delta t \dot{a}_3 \\
\text{do} \quad i_2 = 1, n_2
\]

\[
v = v + \frac{1}{2} n_1 \delta t \dot{a}_2 \\
\text{do} \quad i_1 = 1, n_1
\]

\[
v = v + \frac{1}{2} \delta t \dot{a}_1 \\
r = r + \delta t v \\
calculate forces-1 (i.e. \(a_1\))
\]

\[
v = v + \frac{1}{2} \delta t \dot{a}_1 \\
\text{endo}
\]

\[
calculate forces-2 (i.e. \(a_2\))
\]

\[
v = v + \frac{1}{2} n_1 \delta t \dot{a}_2 \\
\text{endo}
\]

\[
calculate forces-3 (i.e. \(a_3\))
\]

\[
v = v + \frac{1}{2} n_1 n_2 \delta t \dot{a}_3 \\
\text{endo}
\]

\[
calculate forces-4 (i.e. \(a_4\))
\]

\[
v = v + \frac{1}{2} n_1 n_2 n_3 \delta t \dot{a}_4 \\
\text{endo}
\]

In this scheme, \(v\) and \(r\) refer to one of the \(3N\) velocities or positions, respectively. Note that the different types of force are calculated throughout the algorithm. It can be readily seen that the method reduces to the standard velocity Verlet method if \(n_1, n_2\) and \(n_3\) are set equal to 1.

The \(r\)-RESPA method has been applied to a variety of systems, including simple model systems [Tuckerman et al. 1992] but also organic molecules [Watanabe and Karplus 1993], fullerene crystals [Procacci and Berne 1994] and also proteins [Humphreys et al. 1994, 1996]. In these studies the reduction in computational time compared with the standard velocity Verlet method varied between 4-5 and 20-40, depending upon the size of the system, without any noticeable loss in accuracy. Other developments of the \(r\)-RESPA algorithm include its coupling to the fast multipole method (see Section 6.8.3) [Zhou and Berne 1995].

### 7.4 Setting Up and Running a Molecular Dynamics Simulation

In this section we will examine some of the steps involved in performing a molecular dynamics simulation in the microcanonical ensemble. First, it is necessary to establish an
initial configuration of the system. As discussed in Section 6.4.2, the initial configuration may be obtained from experimental data, from a theoretical model or from a combination of the two. It is also necessary to assign initial velocities to the atoms. This can be done by randomly selecting from a Maxwell-Boltzmann distribution at the temperature of interest:

\[
p(v_x) = \left( \frac{m_i}{2\pi k_B T} \right)^{1/2} \exp \left[ -\frac{1}{2} \frac{m_i v_x^2}{k_B T} \right]
\] (7.35)

The Maxwell-Boltzmann equation provides the probability that an atom \( i \) of mass \( m_i \) has a velocity \( v_x \) in the \( x \) direction at a temperature \( T \). A Maxwell-Boltzmann distribution is a Gaussian distribution, which can be obtained using a random number generator. Most random number generators are designed to produce random numbers that are uniform in the range 0 to 1. However, it is relatively straightforward to convert such a random number generator to sample from a Gaussian distribution (or indeed from one of several other distributions [Rubinstein 1981]). The probability of generating a value from a Gaussian (normal) distribution with mean \( \langle x \rangle \) and variance \( \sigma^2 \) \( \sigma^2 = \langle (x - \langle x \rangle)^2 \rangle \) is:

\[
p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[ -\frac{(x - \langle x \rangle)^2}{2\sigma^2} \right]
\] (7.36)

One option is to first generate two random numbers \( \xi_1 \) and \( \xi_2 \) between 0 and 1. The corresponding two numbers from the normal distribution are then calculated using

\[
x_1 = \sqrt{-2\ln \xi_1} \cos(2\pi \xi_2) \quad \text{and} \quad x_2 = \sqrt{-2\ln \xi_2} \sin(2\pi \xi_2)
\] (7.37)

An alternative approach is to generate twelve random numbers \( \xi_1, \ldots, \xi_{12} \) and then calculate:

\[
x = \sum_{i=1}^{12} \xi_i - 6
\] (7.38)

These two methods generate random numbers in the normal distribution with zero mean and unit variance. A number \( x \) generated from this distribution can be related to its counterpart \( x' \) from another Gaussian distribution with mean \( \langle x' \rangle \) and variance \( \sigma \) using

\[
x' = (x' + \sigma x)
\] (7.39)

The initial velocities may also be chosen from a uniform distribution or from a simple Gaussian distribution. In either case the Maxwell-Boltzmann distribution of velocities is usually rapidly achieved.

The initial velocities are often adjusted so that the total momentum of the system is zero. Such a system then samples from the constant NVEP ensemble. To set the total linear momentum of the system to zero, the sum of the components of the atomic momenta along the \( x, y \) and \( z \) axes is calculated. This gives the total momentum of the system in each direction, which, when divided by the total mass, is subtracted from the atomic velocities to give an overall momentum of zero.

Having set up the system and assigned the initial velocities, the simulation proper can commence. At each step the force on each atom must be calculated by differentiating the potential function. The force on an atom may include contributions from the various
terms in the force field such as bonds, angles, torsional terms and non-bonded interactions. The force is straightforward to calculate for two atoms interacting under the Lennard-Jones potential:

\[
f_{ij} = \frac{r_{ij}^2}{|r_{ij}|^3} \frac{24\epsilon}{r_{ij}} \left[ 2 \left( \frac{\sigma}{r_{ij}} \right)^{13} - \left( \frac{\sigma}{r_{ij}} \right)^{7} \right]
\]  

(7.40)

The force between the two atoms is equal in magnitude and opposite in direction and applies along the line connecting the two nuclear centres, in accordance with Newton’s third law (Figure 7.7). It is necessary to calculate the force between each atom pair just once. This is most easily achieved by arranging to compute the force between an atom and those atoms with a higher index (i.e. for an atom \( i \) the forces are calculated with atoms \( i + 1, i + 2, \ldots, N \)). Having calculated the force between an atom \( i \) and an atom with a higher index \( j \), minus the force is added to the accumulating sum of the forces on \( j \). The force calculation is most easily implemented using two loops, as outlined in the following pseudocode:

```plaintext
set elements on force array to zero
while atom1 = 1 to N - 1
    while atom2 = atom1 + 1 to N
        calculate force on atom1 due to interaction with atom2
        add the force to the array element atom1
        subtract the force from the appropriate force array element atom2
    enddo
endo
do
```

At the end of these two loops, the total force on each atom is known. A consequence of the fact that the force between the two atoms is equal and opposite is that the neighbour list for each atom need only contain those atoms with a higher number as the force on an atom due to interactions with lower numbered atoms will be calculated earlier in the loop. This organisation of the neighbour list contrasts with the structure used for Monte Carlo simulations, where all the neighbours of each atom (with both lower and higher indices) must be stored.

Analytical expressions for the forces due to other terms in the molecular mechanics potential function have been published for most of the functional forms encountered in common force
fields. These expressions can seem rather complicated because the intramolecular terms (e.g., bonds, angles, torsions) are calculated in terms of the internal coordinates, whereas molecular dynamics is typically performed using Cartesian coordinates. The chain rule must therefore be used to obtain the desired functional forms. However, the resulting expressions are relatively easy to implement in a computer program.

The first stage of a molecular dynamics simulation is the equilibration phase, the purpose of which is to bring the system to equilibrium from the starting configuration. During equilibration, various parameters are monitored together with the actual configurations. When these parameters achieve stable values then the production phase can commence. It is during the production phase that thermodynamic properties and other data are calculated. The parameters that are used to characterise whether equilibrium has been reached depend to some extent on the system being simulated but invariably include the kinetic, potential and total energies, the velocities, the temperature and the pressure. As we have indicated, the kinetic and potential energies would be expected to fluctuate in a simulation in the microcanonical ensemble but the total energy should remain constant. The components of the velocities should describe a Maxwell–Boltzmann distribution (in all three directions $x$, $y$ and $z$) and the kinetic energy should be equally distributed among the three directions $x$, $y$ and $z$. It is usually desired to perform a simulation at a specified temperature and so it is common practice to adjust the temperature of the system by scaling the velocities (see Section 7.7.1) during the equilibration phase. During the production phase the temperature is a variable of the system. Order parameters may be calculated to monitor changes in structure, which can supplement visual examination of the evolving trajectory.

When simulating an inhomogeneous system a more detailed equilibration procedure is usually desirable. A typical procedure suitable for a molecular dynamics simulation of a macromolecular solute, such as a protein in solution, would be as follows. First, the solvent alone together with any mobile counterions is subject to energy minimisation with the solute kept fixed in its initial conformation. The solvent and any counterions are then allowed to evolve using either a molecular dynamics (or indeed Monte Carlo) simulation, again keeping the structure of the solute molecule fixed. This solvent equilibration phase should be sufficiently extensive to allow the solvent to completely readjust to the potential field of the solute. For molecular dynamics this implies that the length of this solvent equilibration phase should be longer than the relaxation time of the solvent (the time taken for a molecule to lose any ‘memory’ of its original orientation, which for water is about 10 ps). Next, the entire system (solute and solvent) is minimised. Only then does the molecular dynamics simulation of the whole system commence.

At the start of the production phase all counters are set to zero and the system is permitted to evolve. In a microcanonical ensemble no velocity scaling is performed during the production phase and so the temperature becomes a calculated property of the system. Various properties are routinely calculated and stored during the production phase for subsequent analysis and processing. Careful monitoring of these properties during the simulation can show whether the simulation is ‘well behaved’ or not; it may be necessary to restart a simulation if problems are encountered. It is also usual to store the positions, energies
and velocities of configurations at regular intervals (e.g. every five to twenty time steps), from which other properties can be determined once the simulation has finished.

### 7.4.1 Calculating the Temperature

Many thermodynamic properties can be calculated from a molecular dynamics simulation. Most of these were dealt with in Section 6.2; here we just discuss the calculation of temperature. The instantaneous value of the temperature is related to the kinetic energy via the particles’ momenta as follows:

\[
\mathcal{E} = \sum_{i=1}^{N_c} \frac{|p_i|^2}{2m_i} - \frac{k_B T}{2} (3N - N_c)
\]

(7.41)

where \(N_c\) is the number of constraints and so \(3N - N_c\) is the total number of degrees of freedom. For an isolated system (i.e. for a simulation of a system in vacuo) the total translational momentum of the system and the total angular momentum are conserved and can be made equal to zero by an appropriate choice of initial velocities. For a simulation performed using periodic boundary conditions, the total linear momentum is conserved but the total angular momentum is not. It is common practice to choose a set of initial velocities that ensures that the total linear momentum and the total angular momentum are zero. As the system evolves, the linear momentum should remain zero but the angular momentum will not. Molecular dynamics with periodic boundary conditions thus strictly samples from the constant \(NVE\)P ensemble where \(P\) is the total linear momentum. This differs trivially from the standard microcanonical ensemble but it should be remembered that the appropriate number of degrees of freedom must be subtracted from the total when calculating the kinetic energy per degree of freedom. Specifically, for a system in vacuo where the total linear and angular momenta have been set to zero, six degrees of freedom need to be subtracted. For a simulation using periodic boundary conditions three degrees of freedom need to be subtracted if the centre-of-mass motion of the system is removed. In constraint dynamics, discussed in the next section, rather more degrees of freedom may be fixed and \(N_c\) must be calculated accordingly.

### 7.5 Constraint Dynamics

The earliest molecular dynamics simulations using ‘realistic’ potentials were of atoms interacting under the Lennard-Jones potential. In such calculations the only forces on the atoms are those due to non-bonded interactions. It is rather more difficult to simulate molecules because the interaction between two non-spherical molecules depends upon their relative orientation as well as the distance between them. If the molecules are flexible then there will also be intramolecular interactions, which give rise to changes in conformation. Clearly, the simplest model is to treat the species present as rigid bodies with no intramolecular conformational freedom. In such cases the dynamics of each molecule can often be considered in terms of translations of its centre of mass and rotations about its centre of mass. The force on the molecule equals the vector sum of all the forces acting at the
centre of mass, and the rotational motion is determined by the torque about the centre of mass. To deal with these rotational motions is considerably more complicated than for the translational motions, but in favourable cases they can be programmed quite efficiently.

When the simulation involves conformationally flexible molecules then the motion is naturally described in terms of the atomic Cartesian coordinates. The conformational behaviour of a flexible molecule is usually a complex superposition of different motion. 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 molecular dynamics 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 prejudicing the accuracy of the simulation. Constraint dynamics enables individual internal coordinates or combinations of specified coordinate degrees of freedom to be constrained, or ‘fixed’ during the simulation without affecting the other internal degrees of freedom.

Before we consider in detail the use of constraint dynamics, it is helpful to establish the difference between constraints and restraints; we shall discuss the method of restraining molecular dynamics in a later chapter (see Section 9.10). A constraint is a requirement that the system is forced to satisfy. As we shall see, in constraint dynamics bonds or angles are forced to adopt specific values throughout a simulation. When a bond or angle is restrained then it is able to deviate from the desired value; the restraint only acts to ‘encourage’ a particular value. Restraints are most easily incorporated using additional terms in the force field which impose a penalty for deviations from the reference value. An additional difference is that restrained degrees of freedom still have an energy $k_B T$, associated with them, whereas constrained degrees of freedom do not.

The most commonly used method for applying constraints, particularly in molecular dynamics, is the SHAKE procedure of Ryckaert, Cicotti and Berendsen [Ryckaert et al., 1977]. In constraint dynamics the equations of motion are solved while simultaneously satisfying the imposed constraints. Constrained systems have been much studied in classical mechanics; we shall illustrate the general principles using a simple system comprising a box sliding down a frictionless slope in two dimensions (Figure 7.8). The box is constrained to remain on the slope and so the box’s $x$ and $y$ coordinates must always satisfy the equation of the slope (which we shall write as $y = mx + c$). If the slope were not present then the box...
would fall vertically downwards. Constraints are often categorised as *holonomic* or *non-holonomic*. Holonomic constraints can be expressed in the form

\[ f(q_1, q_2, q_3, \ldots, t) = 0 \]  
(7.42)

\[ q_1, q_2, \text{ etc.,} \] are the coordinates of the particles. Non-holonomic constraints cannot be expressed in this way. For example, the motion of a particle constrained to lie on the surface of a sphere is subject to a holonomic constraint, but if the particle is able to fall off the sphere under the influence of gravity then the constraint becomes non-holonomic. A holonomic constraint that keeps a particle on the surface of a sphere can be written:

\[ r^2 - a^2 = 0 \]  
(7.43)

\[ r \] is the distance of the particle from the origin where the sphere of radius \( a \) is centred. The equivalent non-holonomic constraint is written as an inequality:

\[ r^2 - a^2 \geq 0 \]  
(7.44)

SHAKE uses holonomic constraints. 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. A second difficulty is that the magnitude of the constraint forces is unknown. Thus in the case of the box on the slope, the gravitational force acting on the box is in the \( y \) direction whereas the motion is down the slope. The motion is thus not in the same direction as the gravitational force. As such, the total force on the box can be considered to arise from two sources: one due to gravity and the other a constraint force that is perpendicular to the motion of the box (Figure 7.8). As there is no motion perpendicular to the surface of the slope, the constraint force does no work.

As we know, the motion of a system of \( N \) particles can be described in terms of \( 3N \) independent coordinates or degrees of freedom. If there are \( k \) holonomic constraints then the number of degrees of freedom is reduced to \( 3N - k \). It is possible, in principle at least, to find \( 3N - k \) independent coordinates (the *generalised coordinates*), which can then be used to solve the problem directly. For example, the motion of the box can be described using the single coordinate, \( q_s \) along the direction of the slope. The component of the gravitational force that acts along the slope is \( Mg \sin \theta \) and so the acceleration down the slope is \( g \sin \theta \). The position at any time \( t \) can thus be obtained by integrating the following equation of motion:

\[ \frac{d^2 q}{dt^2} = g \sin \theta \]  
(7.45)

The solution to this equation is:

\[ q(t) = q(0) + t\dot{q}(0) + \frac{1}{2} g \sin \theta \]  
(7.46)

where \( q(0) \) is the value of \( q \) at time \( t = 0 \) and \( \dot{q}(0) \) is the initial velocity of the box along the slope. In this simple example it is quite easy to identify the single generalised coordinate that can be used to describe the motion in the constrained system. When there are many constraints, it can be difficult to determine the generalised coordinates. In any case, it is usually desirable to work with the atomic Cartesian coordinates. The motion of the box can be more generally described in terms of the Cartesian \((x, y)\) coordinates of the box as follows.
Newton's equations in the \( x \) and \( y \) directions are:

\[
M \frac{d^2x}{dt^2} = F_{cx} \tag{7.47}
\]

\[
M \frac{d^2y}{dt^2} = -Mg + F_{cy} \tag{7.48}
\]

where \( F_{cx} \) and \( F_{cy} \) are the components of the as yet unknown constraint force in the \( x \) and the \( y \) directions, respectively. We know that the constraint force acts perpendicular to the slope, and so the ratio of its \( x \) and \( y \) components must be:

\[
\frac{F_{cx}}{F_{cy}} = -m \tag{7.49}
\]

The constraint force can be introduced into Newton's equations as a Lagrange multiplier (see Section 1.10.5). To achieve consistency with the usual Lagrangian notation, we write \( F_{cy} \) as \(-\lambda\) and so \( F_{cx} \) equals \( \lambda m \). Thus:

\[
M \frac{d^2x}{dt^2} = \lambda m \tag{7.50}
\]

\[
M \frac{d^2y}{dt^2} = -Mg - \lambda \tag{7.51}
\]

Equations (7.50) and (7.51) contain three unknowns \( (\frac{d^2x}{dt^2}, \frac{d^2y}{dt^2}, \lambda) \). A third equation that links \( x \) and \( y \) is the equation of the slope, which can be written in the following form:

\[
\sigma = mx - y + c = 0 \tag{7.52}
\]

This constraint equation is expressed in terms of \( x \) and \( y \) rather than their second derivatives. However, as \( \sigma(x,y) = 0 \) holds for all \( x, y \), it follows that \( d\sigma = 0 \) and \( d^2\sigma = 0 \) also. Consequently, the constraint equation can be written:

\[
m \frac{d^2x}{dt^2} - \frac{d^2y}{dt^2} = 0 \tag{7.53}
\]

Solving the three equations gives:

\[
\frac{d^2x}{dt^2} = -\frac{m}{1 + m^2} \tag{7.54}
\]

\[
\frac{d^2y}{dt^2} = -\frac{m^2}{1 + m^2} \tag{7.55}
\]

The \( x \) and \( y \) coordinates at time \( t \) are thus given by:

\[
x(t) = x(0) + \frac{dx(0)}{dt} t - \frac{t^2}{2} \frac{m}{(1 + m^2)} \tag{7.56}
\]

\[
y(t) = y(0) + \frac{dy(0)}{dt} t - \frac{t^2}{2} \frac{m^2}{(1 + m^2)} \tag{7.57}
\]
In the general case, the equations of motion for a constrained system involve two types of force: the ’normal’ forces arising from the intra- and intermolecular interactions, and the forces due to the constraints. We are particularly interested in the case where the constraint \( \sigma_k \) requires the bond between atoms \( i \) and \( j \) to remain fixed. The constraint influences the Cartesian coordinates of atoms \( i \) and \( j \). The force due to this constraint can be written as follows:

\[
F_{ci} = \lambda_k \frac{\partial \sigma_k}{\partial x}
\]  

(7.58)

where \( \lambda_k \) is the Lagrange multiplier and \( x \) represents one of the Cartesian coordinates of the two atoms. Applying Equation (7.58) to the above example, we would write \( F_{cx} = \lambda \partial \sigma / \partial x = \lambda \eta \) and \( F_{cy} = \lambda \partial \sigma / \partial y = -\lambda \). If an atom is involved in a number of constraints (because it is involved in more than one constrained bond) then the total constraint force equals the sum of all such terms. The nature of the constraint for a bond between atoms \( i \) and \( j \) is:

\[
\sigma_{ij} = (r_i - r_j)^2 - d_{ij}^2 = 0
\]

(7.59)

The constraint force lies along the bond at all times. For each constrained bond, there is an equal and opposite force on the two atoms that comprise the bond. The overall effect is that the constraint forces do no work. Suppose the constraint \( \eta \) corresponds to the bond length between atoms \( i \) and \( j \). The constraint forces are obtained by differentiating the constraint with respect to the coordinates of atoms \( i \) and \( j \) and multiplying by \( \lambda \) (as yet) undetermined multiplier:

\[
\frac{\partial \sigma_k}{\partial x_k} = 2(r_i - r_j) \quad \text{so} \quad F_{ci} = \lambda (r_i - r_j)
\]

(7.60)

\[
\frac{\partial \sigma_k}{\partial x_j} = -2(r_i - r_j) \quad \text{and} \quad F_{cj} = -\lambda (r_i - r_j)
\]

(7.61)

The factor of 2 that arises when we differentiate the square term has been incorporated into the Lagrange multiplier \( \lambda \). The above expression for the forces can be incorporated into the Verlet algorithm as follows:

\[
r_i(t + \delta t) = 2r_i(t) - r_i(t - \delta t) + \frac{\delta t^2}{m_i} F_i(t) + \sum_k \frac{\lambda_k \delta t^2}{m_i} r_{i_k}(t)
\]

(7.62)

Recall that the positions that would be obtained from the Verlet algorithm without constraints are \( r_i'(t + \delta t) = 2r_i(t) - r_i(t - \delta t) + \delta t^2 F_i(t)/m_i \). The summation in Equation (7.62) is over all constraints \( k \) that affect atom \( i \). These constraints perturb the positions that would otherwise have been obtained from the integration algorithm, and so the above expression can be written:

\[
r_i(t + \delta t) = r_i'(t + \delta t) + \sum_k \frac{\lambda_k \delta t^2}{m_i} r_{i_k}(t)
\]

(7.63)

The next problem is to determine the multipliers \( \lambda_k \) that enable all the constraints to be satisfied simultaneously. This can be done algebraically in simple cases. Suppose we wish to fix the bond in a diatomic molecule. There is just one constraint in this case, and if the
atoms are labelled 1 and 2 we can write:

\[ r_1(t + \delta t) = r_1(t + \delta t) + \lambda_{12}(\delta t^2/m_1)(r_1(t) - r_2(t)) \]  
(7.64)

\[ r_2(t + \delta t) = r_2(t + \delta t) - \lambda_{12}(\delta t^2/m_2)(r_1(t) - r_2(t)) \]  
(7.65)

A third equation is derived from the requirement that the new positions keep the bond at the required distance:

\[ |r_1(t + \delta t) - r_2(t + \delta t)|^2 = |r_1(t) - r_2(t)|^2 = d_{12}^2 \]  
(7.66)

We now have three equations and three unknowns \((r_1(t + \delta t), r_2(t + \delta t)\) and \(\lambda_{12}\)). Subtracting, and putting \(r_{12}(t) = (r_1(t) - r_2(t))\) and \(r'_{12}(t + \delta t) = r_1'(t + \delta t) - r_2'(t + \delta t)\) gives:

\[ r_1(t + \delta t) - r_2(t + \delta t) = r'_{12}(t + \delta t) + \lambda_{12}\delta t^2(1/m_1 + 1/m_2)\cdot r_{12}(t) \]  
(7.67)

Squaring both sides and imposing the constraint gives:

\[ r'_{12}(t + \delta t)^2 + 2\lambda_{12}\delta t^2(1/m_1 + 1/m_2)\cdot r_{12}(t) + \lambda_{12}^2\delta t^4(1/m_1 + 1/m_2)^2\cdot r_{12}(t)^2 = d_{12}^2 \]  
(7.68)

Solving this quadratic equation for \(\lambda_{12}\) enables the new positions \(r_1(t + \delta t)\) and \(r_2(t + \delta t)\) to be determined.

In the case of a triatomic molecule with two bonds (between atoms 1, 2 and 2, 3), two constraint equations are obtained:

\[ r_{12}(t + \delta t) = r'_{12}(t + \delta t) + \delta t^2(1/m_1 + 1/m_2)\lambda_{12}r_{12}(t) - (\delta t^2/m_2)\lambda_{23}r_{23}(t) \]  
(7.69)

\[ r_{23}(t + \delta t) = r'_{23}(t + \delta t) + \delta t^2(1/m_2 + 1/m_3)\lambda_{23}r_{23}(t) - (\delta t^2/m_2)\lambda_{12}r_{12}(t) \]  
(7.70)

These expressions could be solved algebraically but even in this simple case the algebra becomes rather complicated. A solution can be obtained by ignoring the terms that are quadratic in \(\lambda\) as this produces equations which are linear in the Lagrange multipliers \(\lambda\). When there are many constraints, the problem is equivalent to inverting a \(k \times k\) matrix, even when the quadratic terms are ignored. The SHAKE method uses an alternative approach in which 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 tight 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. Another important requirement is that the constrained degrees of freedom should be only weakly coupled to the remaining degrees of freedom, so that the motion of the molecule is not affected by the application of the constraints. The sampling of unconstrained degrees of freedom should also be unaffected. For example, if the bond lengths and angles are constrained in butane then the only degree of freedom remaining is the torsion angle. It is important that this torsion is able to explore its entire range of values in a way that is not biased because of the SHAKE procedure.

Our discussion so far has considered the use of SHAKE with the Verlet algorithm. Versions have also been derived for other integration schemes, such as the leap-frog algorithm, the predictor-corrector methods and the velocity Verlet algorithm. In the case of the velocity Verlet algorithm, the method has been named RATTLE [Anderson 1983].
When velocities appear in the integration algorithm these must be corrected as well as the positions.

Angle constraints can be easily accommodated in the SHAKE scheme by recognizing that an angle constraint simply corresponds to an additional distance constraint. The angle in a triatomic molecule could thus be maintained at the desired value by requiring the distance between the two end atoms to adopt the appropriate value. This is how some small molecules (e.g. water) are maintained in a rigid geometry. For example, the simple point-charge (SPC) model of water uses three distance constraints. However, it is generally accepted that to constrain the bond angles in simulations of conformationally flexible molecules can have a deleterious effect on the efficiency with which the system explores configurational space. This is because many conformational transitions involve some opening or closing of angles as well as rotations about bonds. The most common use of SHAKE is for constraining bonds involving hydrogen atoms due to their much higher vibrational frequencies. This can enable the time step in a molecular dynamics simulation to be increased (e.g. from 1 fs to 2 fs).

The SHAKE method has been extended by Tobias and Brooks [Tobias and Brooks 1988] to enable constraints to be applied to arbitrary internal coordinates. This enables the torsion angle of a rotatable bond to be constrained to a particular value during a molecular dynamics simulation, which is particularly useful when used in conjunction with methods for calculating free energies (see Section 11.7).

### 7.6 Time-dependent Properties

Molecular dynamics generates configurations of the system that are connected in time and so an MD simulation can be used to calculate time-dependent properties. This is a major advantage of molecular dynamics over the Monte Carlo method. Time-dependent properties are often calculated as *time correlation coefficients*.

#### 7.6.1 Correlation Functions

Suppose we have two sets of data values, \( x \) and \( y \), and we wish to determine what correlation (if any) exists between them. For example, imagine that we are performing a simulation of a fluid in a capillary, and that we wish to determine the correlation between the absolute velocity of an atom and its distance from the wall of the tube. One way to do this would be to plot the sets of values as a graph. A correlation function (also known as a correlation coefficient) provides a numerical value that encapsulates the data and quantifies the strength of the correlation. A series of simulations with different capillary diameters could then be compared by examining the correlation coefficients. A variety of correlation functions can be defined, a commonly used one being:

\[
C_{xy} = \frac{1}{M} \sum_{i=1}^{M} x_i y_i = \langle x_i y_i \rangle
\]  

(7.71)
We have assumed that there are $M$ values of $x_i$ and $y_i$ in the data sets. This correlation function can be normalised to a value between $-1$ and $+1$ by dividing by the root-mean-square values of $x$ and $y$:

$$ c_{xy} = \frac{1}{M} \sum_{i=1}^{M} x_i y_i - \frac{\langle x \rangle \langle y \rangle}{\left( \frac{1}{M} \sum_{i=1}^{M} x_i^2 \right) \left( \frac{1}{M} \sum_{i=1}^{M} y_i^2 \right)} \quad \text{(7.72)} $$

A value of 0 indicates no correlation and an absolute value of 1 indicates a high degree of correlation (which may be positive or negative). We will use a lowercase $c$ to indicate a normalised correlation coefficient.

Sometimes the quantities $x$ and $y$ will fluctuate about non-zero mean values ($\langle x \rangle$ and $\langle y \rangle$). Under such circumstances it is typical to consider just the fluctuating part and to define the correlation function as:

$$ c_{xy} = \frac{1}{M} \sum_{i=1}^{M} (x_i - \langle x \rangle)(y_i - \langle y \rangle) - \frac{\langle (x_i - \langle x \rangle)(y_i - \langle y \rangle) \rangle}{\sqrt{\langle (x_i - \langle x \rangle)^2 \rangle \langle (y_i - \langle y \rangle)^2 \rangle}} \quad \text{(7.73)} $$

$c_{xy}$ can also be written in the following useful way:

$$ c_{xy} = \frac{\sum_{i=1}^{M} x_i y_i - \frac{1}{M} \left( \sum_{i=1}^{M} x_i \right) \left( \sum_{i=1}^{M} y_i \right)}{\sqrt{\left[ \sum_{i=1}^{M} x_i^2 - \frac{1}{M} \left( \sum_{i=1}^{M} x_i \right)^2 \right] \left[ \sum_{i=1}^{M} y_i^2 - \frac{1}{M} \left( \sum_{i=1}^{M} y_i \right)^2 \right]}} \quad \text{(7.74)} $$

Equation (7.74) does not require the mean values $\langle x \rangle$ and $\langle y \rangle$ to be determined before the correlation coefficient can be calculated and so values can be accumulated as the simulation proceeds.

A molecular dynamics simulation provides data values at specific times. This enables the value of some property at some instant to be correlated with the value of the same or another property at a later time $t$. The resulting values are known as time correlation coefficients. The correlation function is then written:

$$ C_{xy}(t) = \langle x(t)y(0) \rangle \quad \text{(7.75)} $$

The following two results are useful:

$$ \lim_{t \to 0} C_{xy}(0) = \langle xy \rangle \quad \text{(7.76)} $$

$$ \lim_{t \to \infty} C_{xy}(t) = \langle x \rangle \langle y \rangle \quad \text{(7.77)} $$
If the quantities $x$ and $y$ are different, then the correlation function is sometimes referred to as a cross-correlation function. When $x$ and $y$ are the same then the function is usually called an autocorrelation function. An autocorrelation function indicates the extent to which the system retains a ‘memory’ of its previous values (or, conversely, how long it takes the system to ‘lose’ its memory). A simple example is the velocity autocorrelation coefficient whose value indicates how closely the velocity at a time $t$ is correlated with the velocity at time 0. Some correlation functions can be averaged over all the particles in the system (as can the velocity autocorrelation function) whereas other functions are a property of the entire system (e.g., the dipole moment of the sample). The value of the velocity autocorrelation coefficient can be calculated by averaging over the $N$ atoms in the simulation:

$$C_{\text{ve}}(t) = \frac{1}{N} \sum_{i=1}^{N} \langle v_i(t) \cdot v_i(0) \rangle$$  \hspace{1cm} (7.78)$$

To normalise the function, we divide by $\langle v_i(0) \cdot v_i(0) \rangle$:

$$C_{\text{ve}}(t) = \frac{1}{N} \sum_{i=1}^{N} \frac{\langle v_i(t) \cdot v_i(0) \rangle}{\langle v_i(0) \cdot v_i(0) \rangle}$$  \hspace{1cm} (7.79)$$

In general, an autocorrelation function such as the velocity autocorrelation coefficient has an initial value of 1 and at long times has 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. If $P$ steps of molecular dynamics are required for complete relaxation, and the simulation has been run for a total of $Q$ steps, then $(Q - P)$ different sets of values could be used to calculate a value for the correlation function. The first set would run from step 1 to step $N$; the second from step 2 to step $N + 1$, and so on (Figure 7.9). In fact, as we saw in Section 6.9 the high degree of correlation between successive time steps means that it is common to use time origins that are separated by several time steps, as shown in Figure 7.9. If we use $M$ time origins ($t_j$) then the velocity autocorrelation function is given by:

$$C_{\text{ve}}(t) = \frac{1}{MN} \sum_{j=1}^{M} \sum_{i=1}^{N} v_i(t_j) \cdot v_i(t_j + t)$$  \hspace{1cm} (7.80)$$

**Fig. 7.9**: The use of different time origins improves the accuracy with which time correlation functions can be calculated.
Quantities with small relaxation times can thus be determined with greater statistical precision, as it will be possible to include a greater number of data sets from a given simulation. Moreover, no quantity with a relaxation time greater than the length of the simulation can be determined accurately.

The velocity autocorrelation functions obtained from molecular dynamics simulations of argon at two different densities are shown in Figure 7.10. The correlation coefficient has an initial value of 1 and is then quadratic at short times, a result that is predicted theoretically. The subsequent behaviour depends upon the density of the fluid. For a low-density fluid, the velocity autocorrelation coefficient gradually decreases to zero. At high densities $c_\infty(t)$ crosses the axis and then becomes negative. A negative correlation coefficient 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 neighbours and rebounds, reversing the direction of its motion. At both low density and high density the decay towards zero is significantly slower than the exponential decay predicted by kinetic theory. In fact, the decay varies as $t^{-3/2}$. This was one of the most interesting results obtained from the early molecular dynamics simulations and can be observed even with a hard-sphere model [Alder and Wainwright 1970]. The phenomenon is ascribed to the formation of a 'hydrodynamic vortex'. As the atom moves through the fluid it pushes other atoms out of the way. These atoms circle around and eventually give it a final 'push' so resulting in a less rapid decrease to zero (Figure 7.11).

The slow decay of the velocity autocorrelation function can present practical problems when deriving other properties, such as the transport coefficients, that require the correlation function to be integrated between $t = 0$ and $t = \infty$. The so-called 'long tail' of the autocorrelation function makes a significant contribution to the integral, but unfortunately the statistical uncertainty with which this part of the function can be calculated is greater as fewer segments of the appropriate length can be extracted from the simulation.
The velocity autocorrelation function is an example of a single-particle correlation function, in which the average is calculated not only over time origins but also over all the atoms. Some properties are calculated for the entire system. One such property is the net dipole moment of the system, which is the vector sum of all the individual dipoles of the molecules in the system (clearly the dipole moment of the system can be non-zero only if each individual molecule has a dipole). The magnitude and orientation of the net dipole will change with time and is given by:

\[ \mu_{\text{tot}}(t) = \sum_{i=1}^{N} \mu_i(t) \]  

(7.81)

\( \mu_i(t) \) is the dipole moment of molecule \( i \) at time \( t \). The total dipolar correlation function is given by:

\[ c_{\text{dipole}}(t) = \frac{\langle \mu_{\text{tot}}(t) \cdot \mu_{\text{tot}}(0) \rangle}{\langle \mu_{\text{tot}}(0) \cdot \mu_{\text{tot}}(0) \rangle} \]  

(7.82)

The dipole correlation time of the system is a valuable quantity to calculate as it is related to the sample’s absorption spectrum. Liquids usually absorb in the infrared region of the electromagnetic spectrum, a typical spectrum being shown in Figure 7.12. As can be seen, the spectrum is very broad with none of the sharp peaks characteristic of a well-resolved spectrum for a species in the gas phase. This is because the overall dipole of a liquid does not change at a constant rate but, rather, there is a distribution of frequencies. The intensity of absorption at any frequency depends upon the relative contribution of that frequency to the overall distribution. If, on average, the overall dipole changes very rapidly (i.e. the relaxation time is short) then the maximum in the absorption spectrum will occur at a
higher frequency than if the relaxation time is long. To predict the spectrum from the correlation function it is therefore necessary to extract the relative contribution of each dipole fluctuation. This is done using Fourier analysis techniques, in which the correlation function is transformed from the time domain into the frequency domain (an introduction to Fourier analysis is provided in Section 1.10.8). The Fourier analysis picks out the intensity of dipole fluctuation at each frequency using the following relationship:

$$
\hat{c}_{\text{dipole}}(\nu) = \int_{-\infty}^{\infty} c_{\text{dipole}}(t) \exp(-i2\pi\nu t) \, dt
$$  \hspace{1cm} (7.83)

Having calculated the Fourier transform it is then possible to plot the simulated spectrum and compare it to that obtained by experiment, as in Figure 7.12.

### 7.6.2 Orientational Correlation Functions

Other orientational correlation coefficients can be calculated in the same way as the correlation coefficients that we have discussed already. Thus, the reorientational correlation coefficient of a single rigid molecule indicates the degree to which the orientation of a molecule at a time $t$ is related to its orientation at time 0. The angular velocity autocorrelation function is the rotational equivalent of the velocity correlation function:

$$
c_{\omega}(t) = \frac{\langle \omega(0) \cdot \omega(t) \rangle}{\langle \omega(0) \cdot \omega(0) \rangle}
$$  \hspace{1cm} (7.84)
In a liquid, the rotation of a molecule is influenced by neighbouring molecules and over time the correlation will decay to zero. The information embodied in the orientational correlation functions can be compared to a variety of spectroscopic experiments, including infrared, Raman and NMR spectra. For non-spherical molecules it can be useful to derive separate autocorrelation functions for the angular velocity along each of the principal axes of rotation. For example, for a spherical molecule such as CBr₄, neighbouring molecules have a relatively small influence on the loss in correlation in the angular velocity. By contrast, a linear molecule such as CS₂ experiences significant torques as it rotates. This has the effect of damping the rotational motion more severely than for the spherical case, and indeed the correlation function can change sign, indicating that the molecule is now rotating in the opposite direction. For some molecules such as water the presence of specific interactions between molecules (for example, due to hydrogen bonding) can give rise to very rapid damping and several minima in cₜₛ(t).

### 7.6.3 Transport Properties

Transport refers to a phenomenon that gives rise to a flow of material from one region to another. For example, if a solution is prepared with a non-equilibrium solute distribution, then the solute diffuses until the concentration is equal throughout. If a thermal gradient is created, energy flows until the temperature is equalised. A momentum gradient gives rise to viscosity. The very existence of transport implies that the system is not in equilibrium. Techniques have been developed to perform non-equilibrium molecular dynamics simulations from which transport properties can be calculated, but we will not consider these here. We are thus faced with the problem of calculating non-equilibrium properties from equilibrium simulations. This may seem an impossible task but can be achieved by considering the microscopic local fluctuations that occur even in systems at equilibrium. We should be aware, however, that non-equilibrium molecular dynamics simulations can be a more efficient way to calculate transport properties and other quantities [Allen and Tildesley 1987].

To a first approximation the rate at which transport of the relevant quantity occurs (called the flux) is proportional to the gradient of the property with the constant of proportionality being the relevant transport property coefficient. For example, the flux of matter Jₙ (i.e. the amount passing through unit area in unit time) equals the diffusion coefficient (D) multiplied by the concentration gradient; this is Fick's first law of diffusion:

\[ Jₙ = \frac{dN}{dz} \]  
(7.85)

\(N\) is the number density (the number of atoms per unit volume). Equation (7.85) refers to diffusion in the z direction. The minus sign indicates that flux increases in the direction of negative concentration gradient. The time dependence of diffusive behaviour (which applies if a distribution is established at some time and is then allowed to evolve) is governed by Fick's second law, which gives the rate of change of concentration with time:

\[ \frac{\partial N(z,t)}{\partial t} = D \frac{\partial^2 N(z,t)}{\partial z^2} \]  
(7.86)

To solve Fick's second law equation it is necessary to impose two boundary conditions for the spatial dependence and one boundary condition for the temporal dependence (the
equation is second order in space and first order in time). For example, we might require that at time zero all \(N_0\) particles have \(z = 0\). The solution to the equation is then:

\[
\mathcal{N}(z,t) = \frac{N_0}{A\sqrt{\pi Dt}} \exp \left[ -\frac{z^2}{4Dt} \right]
\]  

(7.87)

where \(A\) is the cross-sectional area of the sample. Equation (7.87) is a Gaussian function which initially has a sharp peak at \(z = 0\) but which gradually becomes more spread out as time progresses. When the material being simulated is a pure liquid then the coefficient \(D\) is often referred to as a self-diffusion coefficient. The diffusion coefficient is related to the mean square distance, \(<|r(t) - r(0)|^2>\), which Einstein showed was equal to \(2Dt\). In three dimensions, the mean square displacement is given by:

\[
3D = \lim_{t \to \infty} \frac{<|r(t) - r(0)|^2>}{2t}
\]  

(7.88)

As indicated, the relationship strictly holds only in the limit as \(t \to \infty\).

The Einstein relationship can thus be used to calculate the diffusion coefficient from an equilibrium simulation, by plotting the mean square displacement as a function of time and then attempting to obtain the limiting behaviour as \(t \to \infty\) (Fick's law is inapplicable at short times). The quantity \(<|r(t) - r(0)|>\) can be averaged over the particles in the system to reduce the statistical error. It is also usual practice to average over time origins where possible. When using this method for calculating the diffusion coefficient the mean-squared distances should not be limited by the edges of the periodic box. In other words, we require a set of positions that have not been translated back into the central simulation cell. This can be achieved either by storing a set of 'uncorrected' positions or indeed by not correcting any of the positions during the simulation and simply generating the appropriate minimum image positions as required for the calculation of the energy or forces.

Einstein relationships hold for other transport properties, e.g. the shear viscosity, the bulk viscosity and the thermal conductivity. For example, the shear viscosity \(\eta\) is given by:

\[
\eta_{xy} = \frac{1}{Vk_B T} \lim_{t \to \infty} \frac{<\sum_{i=1}^{N} m \dot{x}_i(t) \dot{y}_i(t) - \sum_{i=1}^{N} \dot{m} \dot{x}_i(t) \dot{y}_i(t)>}{2t}
\]  

(7.89)

The shear viscosity is a tensor quantity, with components \(\eta_{xy}, \eta_{zx}, \eta_{yz}, \eta_{yx}, \eta_{yz}, \eta_{zx}\). It is a property of the whole sample rather than of individual atoms and so cannot be calculated with the same accuracy as the self-diffusion coefficient. For a homogeneous fluid the components of the shear viscosity should all be equal and so the statistical error can be reduced by averaging over the six components. An estimate of the precision of the calculation can then be determined by evaluating the standard deviation of these components from the average. Unfortunately, Equation (7.89) cannot be directly used in periodic systems, even if the positions have been unfolded, because the 'unfolded' distance between two particles may not correspond to the distance of the minimum image that is used to calculate the force. For this reason alternative approaches are required.

One alternative approach to the calculation of the diffusion and other transport coefficients is via an appropriate autocorrelation function. For example, the diffusion coefficient
depends upon the way in which the atomic position \( r(t) \) changes with time. At a time \( t \) the difference between \( r(t) \) and \( r(0) \) is given by:

\[
|r(t) - r(0)| = \int_0^t v(t') \, dt'
\]  

(7.90)

If both sides of Equation (7.90) are now squared then we obtain the following for the mean-square value:

\[
\langle |r(t) - r(0)|^2 \rangle = \int_0^t dt' \int_0^t dt'' (v(t') \cdot v(t''))
\]  

(7.91)

The crucial feature to recognise is that the relevant correlation functions are unaffected by changing the origin, which means that the following holds:

\[
\langle v(t') \cdot v(t'') \rangle = \langle v(t'' - t') \cdot v(0) \rangle
\]  

(7.92)

Integration of the double integral, Equation (7.91) leads to the Green–Kubo formula:

\[
\frac{\langle |r(t) - r(0)|^2 \rangle}{2t} = \int_0^\tau \langle v(\tau) \cdot v(0) \rangle \left(1 - \frac{\tau}{t}\right) \, d\tau
\]  

(7.93)

In the limit:

\[
\int_0^\infty \langle v(\tau) \cdot v(0) \rangle \, d\tau = \lim_{t \to \infty} \frac{\langle |r(t) - r(0)|^2 \rangle}{2t} = 3D
\]  

(7.94)

We can now see why long time-tails in the autocorrelation functions are so important. The area under the curve during the slow decay towards zero may be a significant part of the integral in the Green–Kubo formula. In practice, these integrals are determined numerically. The long time-tail may be dealt with by fitting a function to the curve and then attempting to integrate to infinity.

### 7.7 Molecular Dynamics at Constant Temperature and Pressure

Molecular dynamics is traditionally performed in the constant \( NVE \) (or \( NVEP \)) ensemble. Although thermodynamic results can be transformed between ensembles, this is strictly only possible in the limit of infinite system size (the thermodynamic limit). It may therefore be desired to perform the simulation in a different ensemble. The two most common alternative ensembles are the constant \( NVT \) and the constant \( NPT \) ensembles. In this section we will therefore consider how molecular dynamics simulations can be performed under conditions of constant temperature and/or constant pressure.

#### 7.7.1 Constant Temperature Dynamics

There are several reasons why we might want to maintain or otherwise control the temperature during a molecular dynamics simulation. Even in a constant \( NVE \) simulation it is common practice to adjust the temperature to the desired value during the equilibration phase. A constant temperature simulation may be required if we wish to determine how
the behaviour of the system changes with temperature, such as the unfolding of a protein or glass formation. Simulated annealing algorithms require the temperature of the system to be reduced gradually while the system explores its degrees of freedom. Simulated annealing is used in searching conformational space and in the elucidation of macromolecular structure from NMR and X-ray data and is discussed in Section 9.9.2.

The temperature of the system is related to the time average of the kinetic energy, which for an unconstrained system is given by:

\[ \langle KE \rangle_{NVT} = \frac{3}{2} N k_B T \]  

(7.95)

An obvious way to alter the temperature of the system is thus to scale the velocities [Woodcock 1971]. If the temperature at time \( t \) is \( T(t) \) and the velocities are multiplied by a factor \( \lambda \), then the associated temperature change can be calculated as follows:

\[ \Delta T = \frac{1}{2} \sum_{i=1}^{N} \frac{2 m_i (\lambda \dot{v}_i)^2}{N k_B} - \frac{1}{2} \sum_{i=1}^{N} \frac{2 m_i \dot{v}_i^2}{N k_B} \]  

(7.96)

\[ \Delta T = (\lambda^2 - 1) T(t) \]  

(7.97)

\[ \lambda = \sqrt{T_{new}/T(t)} \]  

(7.98)

The simplest way to control the temperature is thus to multiply the velocities at each time step by the factor \( \lambda = \sqrt{T_{new}/T_{curr}} \), where \( T_{curr} \) is the current temperature as calculated from the kinetic energy and \( T_{new} \) is the desired temperature.

An alternative way to maintain the temperature is to couple the system to an external heat bath that is fixed at the desired temperature [Berendsen et al. 1984]. The bath acts as a source of thermal energy, supplying or removing heat from the system as appropriate. The velocities are scaled at each step, such that the rate of change of temperature is proportional to the difference in temperature between the bath and the system:

\[ \frac{dT(t)}{dt} = \frac{1}{\tau} (T_{bath} - T(t)) \]  

(7.99)

\( \tau \) is a coupling parameter whose magnitude determines how tightly the bath and the system are coupled together. This method gives an exponential decay of the system towards the desired temperature. The change in temperature between successive time steps is:

\[ \Delta T = \frac{\delta t}{\tau} (T_{bath} - T(t)) \]  

(7.100)

The scaling factor for the velocities is thus:

\[ \lambda^2 = 1 + \frac{\delta t}{\tau} \left( \frac{T_{bath}}{T(t)} - 1 \right) \]  

(7.101)

If \( \tau \) is large, then the coupling will be weak. If \( \tau \) is small, the coupling will be strong and when the coupling parameter equals the time step \( (\tau = \delta t) \) then the algorithm is equivalent to the simple velocity scaling method. A coupling constant of approximately 0.4 ps has been suggested as an appropriate value to use when the time step is 1 fs, giving \( \delta t/\tau \approx 0.0025 \). The advantage of this approach is that it does permit the system to fluctuate about the desired temperature.
These two relatively simple temperature scaling methods do not generate rigorous canonical averages. Velocity scaling artificially prolongs any temperature differences among the components of the system, which can lead to the phenomenon of 'hot solvent, cold solute', in which the 'temperature' of the solute is lower than that of the solvent, even though the overall temperature of the system is at the desired value. One 'solution' to this problem is to apply temperature coupling separately to the solute and to the solvent but the problem of unequal distribution of energy between the various components (and between the various modes of motion) may still remain. Two methods that do generate rigorous canonical ensembles if properly implemented are the stochastic collisions method and the extended system method.

In the stochastic collisions method a particle is randomly chosen at intervals and its velocity is reassigned by random selection from the Maxwell-Boltzmann distribution [Anderson 1980]. This is equivalent to the system being in contact with a heat bath that randomly emits 'thermal particles' which collide with the atoms in the system. Between each collision the system is simulated at constant energy and so the overall effect is equivalent to a series of microcanonical simulations, each performed at a slightly different energy. The distribution of the energies of these 'mini microcanonical' simulations will be a Gaussian function. The stochastic collisions method does not, of course, generate a smooth trajectory, which may be a drawback. By calculating the energy change due to a collision, Anderson showed that the mean rate \( \nu \) at which each particle should suffer a stochastic collision is given by:

\[
\nu = \frac{2a\kappa}{3k_B\mathcal{N}^{1/3}N^{2/3}}
\]  

(7.102)

\( a \) is a dimensionless constant, \( \kappa \) is the thermal conductivity and \( \mathcal{N} \) is the number density of the particles. If the thermal conductivity is not known then a suitable value of \( \nu \) can be obtained from the intermolecular collision frequency \( \nu_c \):

\[
\nu = \nu_c/N^{2/3}
\]  

(7.103)

If the collision rate is too low then the system does not sample from a canonical distribution of energies. If it is too high then the temperature control algorithm dominates and the system does not show the expected fluctuations in kinetic energy. The velocity of more than one particle can be changed in the stochastic collision method; in the limit the velocities of all the particles are changed simultaneously, though it is preferable to do this at quite long intervals. A distinction can thus be made between 'minor' collisions, in which only one (or a few) particles are affected, and 'major' (or 'massive') collisions, where the velocities of all particles are changed. It is also possible to use a combined approach, with minor collisions occurring relatively frequently and major collisions at longer intervals.

Extended system methods, originally introduced for performing constant temperature molecular dynamics by Nosé [Nosé 1984] and subsequently developed by Hoover [Hoover 1985], consider the thermal reservoir to be an integral part of the system. The reservoir is represented by an additional degree of freedom, labelled \( s \). The reservoir has potential energy \( (f + 1)k_BT \ln s \), where \( f \) is the number of degrees of freedom in the physical system and \( T \) is the desired temperature. The reservoir also has kinetic energy \( (Q/2)(ds/dt)^2 \), \( Q \) is a parameter with the dimensions of energy \( \times \) (time)\(^2\) and can be considered the
(fictitious) mass of the extra degree of freedom. The magnitude of \( Q \) determines the coupling between the reservoir and the real system and so influences the temperature fluctuations.

Each state of the extended system that is generated by the molecular dynamics simulation corresponds to a unique state of the real system. There is not, however, a direct correspondence between the velocities and the time in the real and the extended systems. The velocities of the atoms in the real system are given by:

\[
v_i = s \frac{dr_i}{dt} \tag{7.104}
\]

\( r_i \) is the position of particle \( i \) in the simulation and \( v_i \) is considered to be the real velocity of the particle. The time step \( \delta t \) is related to the time step in 'real time' \( dt \) by

\[
\delta t = s \delta t' \tag{7.105}
\]

The value of the additional degree of freedom \( s \) can change and so the time step in real time can fluctuate. Thus regular time intervals in the extended system correspond to a trajectory of the real system which is unevenly spaced in time.

The parameter \( Q \) controls the energy flow between the system and the reservoir. If \( Q \) is large, then the energy flow is slow; in the limit of infinite \( Q \), conventional molecular dynamics is regained and there is no energy exchange between the reservoir and the real system. However, if \( Q \) is too small then the energy oscillates, resulting in equilibration problems. Nose has suggested that \( Q \) should be proportional to \( f k_B T \); the constant of proportionality can then be obtained by performing a series of trial simulations for a test system and observing how well the system maintains the desired temperature.

### 7.7.2 Constant Pressure Dynamics

Just as one may wish to specify the temperature in a molecular dynamics simulation, so it may be desired to maintain the system at a constant pressure. This enables the behaviour of the system to be explored as a function of the pressure, enabling one to study phenomena such as the onset of pressure-induced phase transitions. Many experimental measurements are made under conditions of constant temperature and pressure, and so simulations in the isothermal-isobaric ensemble are most directly relevant to experimental data. Certain structural rearrangements may be achieved more easily in an isobaric simulation than in a simulation at constant volume. Constant pressure conditions may also be important when the number of particles in the system changes (as in some of the 'test particle' methods for calculating free energies and chemical potentials; see Section 8.9).

The pressure often fluctuates much more than quantities such as the total energy in a constant \( NVE \) molecular dynamics simulation. This is as expected because the pressure is related to the virial, which is obtained as the product of the positions and the derivative of the potential energy function. This product, \( r_i \frac{dv}{dr_j} (r_j) \frac{dr_j}{dr_i} \), changes more quickly with \( i \) than does the internal energy, hence the greater fluctuation in the pressure.

A macroscopic system maintains constant pressure by changing its volume. A simulation is the isothermal-isobaric ensemble also maintains constant pressure by changing the volume.
of the simulation cell. The amount of volume fluctuation is related to the isothermal compressibility, \( \kappa \):

\[
\kappa = -\frac{1}{V} \left( \frac{\partial V}{\partial P} \right) _T
\]  

(7.1)

An easily compressible substance has a larger value of \( \kappa \), so larger volume fluctuations occur at a given pressure than in a more incompressible substance. Conversely, in a constant volume simulation a less compressible substance shows larger fluctuations in the pressure. The isothermal compressibility is the pressure analogue of the heat capacity, which is related to the energy fluctuations.

A volume change in an isobaric simulation can be achieved by changing the volume in directions, or in just one direction. It is instructive to consider the range of volume changes that one might expect to observe in a constant pressure simulation of a 'typical' system. The isothermal compressibility is related to the mean square volume displacement by:

\[
\kappa = \frac{1}{k_BT} \frac{\langle V^2 \rangle - \langle V \rangle ^2}{\langle V^2 \rangle} 
\]  

(7.10)

The isothermal compressibility of an ideal gas is approximately 1 atm\(^{-1}\). So for a simulation in a box of side 20 Å (volume 8000 Å\(^3\)) at 300 K, the root mean square change in the volume is approximately 18100 Å\(^2\). This is larger than the initial size of the box! For a relative incompressible substance such as water (\( \kappa = 44.75 \times 10^{-6} \) atm\(^{-1}\)) the fluctuation is 121 Å, which corresponds to the box only changing by about 0.1 Å in each direction. The values have clear implications for the appropriate size of the simulation system.

Many of the methods used for pressure control are analogous to those used for temperature control. Thus, the pressure can be maintained at a constant value by simply scaling the volume. An alternative is to couple the system to a 'pressure bath', analogous to a temperature bath [Berendsen et al. 1984]. The rate of change of pressure is given by:

\[
\frac{dP(t)}{dt} = \frac{1}{\tau_p} (P_{\text{bath}} - P(t))
\]  

(7.10)

\( \tau_p \) is the coupling constant, \( P_{\text{bath}} \) is the pressure of the 'bath', and \( P(t) \) is the actual pressure at time \( t \). The volume of the simulation box is scaled by a factor \( \lambda \), which is equivalent to scaling the atomic coordinates by a factor \( \lambda^{1/3} \). Thus:

\[
\lambda = 1 - \kappa \frac{\delta t}{\tau_p} (P - P_{\text{bath}})
\]  

(7.10)

The new positions are given by:

\[
x'_i = \lambda^{1/3} x_i
\]  

(7.11)

The constant \( \kappa \) can be combined with the relaxation constant \( \tau_p \) as a single constant. The expression can be applied isotropically (i.e. such that the scaling factor is equal for all directions) or anisotropically (where the scaling factor is calculated independently for each of the three axes). In general, it is best to use the anisotropic approach as it enables the box dimensions to change independently. Unfortunately, it has not been possible to determine from which ensemble this method samples.
In the extended pressure-coupling system methods, first introduced by Anderson [Anderson 1980], an extra degree of freedom, corresponding to the volume of the box, is added to the system. The kinetic energy associated with this degree of freedom (which can be considered to be equivalent to a piston acting on the system) is \( \frac{1}{2} Q (dV/dt)^2 \), where \( Q \) is the 'mass' of the piston. The piston also has potential energy \( PV \), where \( P \) is the desired pressure and \( V \) is the volume of the system. A piston of small mass gives rise to rapid oscillations in the box, whereas a large mass has the opposite effect. An infinite mass returns normal molecular dynamics behaviour. The volume can vary during the simulation, with the average volume being determined by the balance between the internal pressure of the system and the desired external pressure. The extended-system temperature-scaling method of Nosé uses a scaled time; in the extended pressure method the coordinates of the extended system are related to the 'real' coordinates by:

\[
x'_i = V^{-1/3} r_i
\]

(7.111)

### 7.8 Incorporating Solvent Effects into Molecular Dynamics: Potentials of Mean Force and Stochastic Dynamics

In many simulations of solute-solvent systems the primary focus is the behaviour of the solute; the solvent is of relatively little interest, particularly in regions far from the solute molecule. The use of non-rectangular periodic boundary conditions, stochastic boundaries and 'solvent shells' can all help to reduce the number of solvent molecules required and enable a larger proportion of the computing time to be spent simulating the solute. In this section we consider a group of techniques that incorporate the effects of solvent without requiring any explicit specific solvent molecules to be present.

One approach to this problem is to use a potential of mean force (PMF), which describes how the free energy changes as a particular coordinate (such as the separation of two atoms or the torsion angle of a bond) is varied. The free energy change described by the potential of mean force includes the averaged effects of the solvent.

Potentials of mean force may be determined using a molecular dynamics or Monte Carlo simulation using the techniques of umbrella sampling or free energy perturbation, which will be discussed in Chapter 11. Here we illustrate the concept using an example. The energy difference between the \textit{trans} and \textit{gauche} conformations for an isolated molecule of 1,2-dichloroethane (i.e. in the gas phase) is approximately 1.14 kcal mol\(^{-1}\) with a population containing 77\% \textit{trans} and 23\% \textit{gauche} conformers. In liquid 1,2-dichloroethane, however, the relative population of the \textit{gauche} conformer is significantly increased relative to the \textit{trans} conformer by comparison with the isolated molecule, with 44\% \textit{trans} and 56\% \textit{gauche}. These experimental results were reproduced by Jorgensen (see Figure 7.13) using Monte Carlo simulations [Jorgensen et al. 1981]. The potential of mean force would be designed to reproduce this new population and so enable a single 1,2-dichloroethane molecule to be simulated as if it were present in the liquid.

A simulation performed using a potential of mean force enables the modulating effects of the solvent to be taken into account. The solvent also influences the dynamic behaviour of the