3

Molecular dynamics

3.1 Equations of motion for atomic systems

In this chapter, we deal with the techniques used to solve the classical equations of motion for a system of N molecules interacting via a potential $\mathcal{V}$ as in eqn (1.4). These equations may be written down in various ways (Goldstein, 1980). Perhaps the most fundamental form is the Lagrangian equation of motion

$$\frac{\mathrm{d}}{\mathrm{d}t}(\partial\mathcal{L}/\partial\dot{q}_k) - (\partial\mathcal{L}/\partial q_k) = 0 \tag{3.1}$$

where the Lagrangian function $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$ is defined in terms of kinetic and potential energies

$$\mathcal{L} = \mathcal{K} - \mathcal{V} \tag{3.2}$$

and is considered to be a function of the generalized coordinates q_k and their time derivatives $\dot{q}_k$. If we consider a system of atoms, with Cartesian coordinates $\mathbf{r}_i$ and the usual definitions of $\mathcal{K}$ and $\mathcal{V}$ (eqns (1.3) and (1.4)) then eqn (3.1) becomes

$$m_i\ddot{\mathbf{r}}_i = \mathbf{f}_i \tag{3.3}$$

where m_i is the mass of atom i and

$$\mathbf{f}_i = \boldsymbol{\nabla}_{\mathbf{r}_i}\mathcal{L} = -\boldsymbol{\nabla}_{\mathbf{r}_i}\mathcal{V} \tag{3.4}$$

is the force on that atom. These equations also apply to the centre of mass motion of a molecule, with $\mathbf{f}_i$ representing the total force on molecule i; the equations for rotational motion may also be expressed in the form of eqn (3.1), and will be dealt with in Sections 3.3 and 3.4.

The generalized momentum p_k conjugate to q_k is defined as

$$p_k = \partial\mathcal{L}/\partial\dot{q}_k. \tag{3.5}$$

The momenta feature in the Hamiltonian form of the equations of motion

$$\dot{q}_k = \partial\mathcal{H}/\partial p_k \tag{3.6a}$$

$$\dot{p}_k = -\partial\mathcal{H}/\partial q_k. \tag{3.6b}$$

Computer Simulation of Liquids. Second Edition. M. P. Allen and D. J. Tildesley.

The Hamiltonian is strictly defined by the equation

$$\mathcal{H}(\mathbf{p},\,\mathbf{q}) = \sum_k \dot{q}_k p_k - \mathcal{L}(\mathbf{q},\,\dot{\mathbf{q}}) \tag{3.7}$$

where it is assumed that we can write $\dot{q}_k$ on the right as some function of the momenta $\mathbf{p}$. For our immediate purposes (involving a potential $\mathcal{V}$ which is independent of velocities and time) this reduces to eqn (1.2), and $\mathcal{H}$ is automatically equal to the energy (Goldstein, 1980, Chapter 8). For Cartesian coordinates, Hamilton's equations become

$$\dot{\mathbf{r}}_i = \mathbf{p}_i/m_i \tag{3.8a}$$

$$\dot{\mathbf{p}}_i = -\nabla_{\mathbf{r}_i}\mathcal{V} = \mathbf{f}_i. \tag{3.8b}$$

Computing centre of mass trajectories, then, involves solving either a system of $3N$ second-order differential equations, eqn (3.3), or an equivalent set of $6N$ first-order differential equations, eqns (3.8a), (3.8b). Before considering how to do this, we can make some very general remarks regarding the equations themselves.

A consequence of eqn (3.6b), or equivalently eqns (3.5) and (3.1), is that in certain circumstances a particular generalized momentum p_k may be conserved, that is, $\dot{p}_k = 0$. The requirement is that $\mathcal{L}$, and hence $\mathcal{H}$ in this case, shall be independent of the corresponding generalized coordinate q_k. For any set of particles, it is possible to choose six generalized coordinates, changes in which correspond to translations of the centre of mass, and rotations about the centre of mass, for the system as a whole (changes in the remaining $3N - 6$ coordinates involving motion of the particles relative to one another). If the potential function $\mathcal{V}$ depends only on the magnitudes of particle separations (as is usual) and there is no external field applied (i.e. the term v_1 in eqn (1.4) is absent) then $\mathcal{V}$, $\mathcal{H}$ and $\mathcal{L}$ are manifestly independent of these six generalized coordinates. The corresponding conjugate momenta, in Cartesian coordinates, are the total linear momentum

$$\mathbf{P} = \sum_i \mathbf{p}_i \tag{3.9}$$

and the total angular momentum

$$\mathbf{L} = \sum_i \mathbf{r}_i \times \mathbf{p}_i = \sum_i m_i \mathbf{r}_i \times \dot{\mathbf{r}}_i \tag{3.10}$$

where we take the origin at the centre of mass of the system. Thus, these are conserved quantities for a completely isolated set of interacting molecules. In practice, we rarely consider completely isolated systems. A more general criterion for the existence of these conservation laws is provided by symmetry considerations (Goldstein, 1980, Chapter 8). If the system (i.e. $\mathcal{H}$) is invariant to translation in a particular direction, then the corresponding momentum component is conserved. If the system is invariant to rotation about an axis, then the corresponding angular momentum component is conserved. Thus, we occasionally encounter systems enclosed in a spherical box, and so a spherically symmetrical v_1 term appears in eqn (1.4); all three components of total angular momentum about the centre of symmetry will be conserved, but total translational momentum will not be. If the surrounding walls formed a cubical box, none of these quantities would be conserved.

In the case of the periodic boundary conditions described in Chapter 1, it is easy to see that translational invariance is preserved, and hence total linear momentum is conserved. Several different box geometries were considered in Chapter 1, but none of them were spherically symmetrical; in fact it is impossible (in Euclidean space) to construct a spherically symmetric periodic system. Hence, despite the fact that there may be no v_1-term in eqn (1.4), total angular momentum is not conserved in most molecular dynamics simulations. In the case of the spherical boundary conditions discussed in Section 1.6.5, a kind of angular momentum conservation law does apply. When we embed a two-dimensional system in the surface of a sphere, the three-dimensional spherical symmetry is preserved. Similarly, for a three-dimensional system, there should be a four-dimensional conserved 'hyper-angular momentum'.

We have left until last the most important conservation law. Assuming that $\mathcal{H}$ does not depend explicitly on time (so that $\partial\mathcal{H}/\partial t = 0$), the total derivative $\dot{\mathcal{H}}$ may be written

$$\frac{\mathrm{d}\mathcal{H}}{\mathrm{d}t} = \sum_k \left(\frac{\partial \mathcal{H}}{\partial q_k} \dot{q}_k + \frac{\partial \mathcal{H}}{\partial p_k} \dot{p}_k \right) = 0$$

by virtue of eqns (3.6). Hence the Hamiltonian is a constant of the motion. This energy conservation law applies whether or not an external potential exists: the essential condition is that no explicitly time-dependent (or velocity-dependent) forces shall act on the system.

The second point concerning the equations of motion is that they are reversible in time. By changing the signs of all the velocities or momenta, we will cause the molecules to retrace their trajectories. If the equations of motion are solved correctly, the computer-generated trajectories will also have this property.

Our final observation concerning eqns (3.3), (3.4), and (3.6) is that the spatial derivative of the potential appears. This leads to a qualitative difference in the form of the motion, and the way in which the equations are solved, depending upon whether or not $\mathcal{V}$ is a continuous function of particle positions. To use the finite-timestep method of solution to be described in the next section, it is essential that the particle positions vary smoothly with time: a Taylor expansion of $\mathbf{r}(t)$ about time t may be necessary, for example. Whenever the potential varies sharply (as in the hard-sphere and square-well cases) impulsive 'collisions' between particles occur at which the velocities (typically) change discontinuously. The particle dynamics at the moment of each collision must be treated explicitly, and separately from the smooth inter-collisional motion. The identification of successive collisions is the key feature of a molecular dynamics program for such systems, and we shall discuss this in Section 3.7.

3.2 Finite-difference methods

A standard method for solution of ordinary differential equations such as eqns (3.3) and (3.8) is the finite-difference approach. The general idea is as follows. Given the molecular positions, velocities, and other dynamic information at time t, we attempt to obtain the positions, velocities, etc. at a later time $t + \delta t$, to a sufficient degree of accuracy. The equations are solved on a step-by-step basis; the choice of the time interval δt will depend somewhat on the method of solution, but δt will be significantly smaller than

the typical time taken for a molecule to travel its own length. Many different algorithms fall into the general finite-difference pattern. Historically, standard approaches such as predictor–corrector algorithms (Gear, 1966; 1971) and general-purpose approaches such as Runge–Kutta (Press et al., 2007) have been used in molecular dynamics simulations, and there have been several comparisons of different methods (Gear, 1971; van Gunsteren and Berendsen, 1977; Hockney and Eastwood, 1988; Berendsen and van Gunsteren, 1986; Gray et al., 1994; Leimkuhler and Reich, 2004). Which shall we choose?

A shortlist of desirable qualities for a successful simulation algorithm might be as follows.

(a) It should be fast, and require little memory.

(b) It should permit the use of a long timestep δt.

(c) It should duplicate the classical trajectory as closely as possible.

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

(e) It should be simple in form and easy to program.

For molecular dynamics, the first point is generally less critical than the others, when it comes to choosing between algorithms. The memory required to store positions, velocities, accelerations, etc. is very small compared with that available on most computers, although this might become a consideration when taking advantage of special features of the architecture, such as graphics processing units (GPUs). Compared with the time-consuming force calculation, which is carried out at every timestep, the raw speed of the integration algorithm is not crucial. It is far more important to be able to employ a long timestep δt: in this way, a given period of 'simulation' time can be covered in a modest number of integration steps, that is, in an acceptable amount of computer time. Clearly, the larger δt, the less accurately will our solution follow the correct classical trajectory. How important are points (c) and (d) in the list?

It is unreasonable to expect that any approximate method of solution will dutifully follow the exact classical trajectory indefinitely. Any two classical trajectories which are initially very close will eventually diverge from one another exponentially with time (according to the 'Lyapunov exponents'), irrespective of the algorithm used to approximate the equations of motion. In the same way, any small perturbation, even the tiny error associated with finite precision arithmetic, will tend to cause a computer-generated trajectory to diverge from the true classical trajectory with which it is initially coincident. We illustrate the effect in Fig. 3.1: using one simulation as a reference, we show that a small perturbation applied at time $t = 0$ causes the trajectories in the perturbed simulation to diverge from the reference trajectories and become statistically uncorrelated, within a few hundred timesteps (see also Stoddard and Ford, 1973; Erpenbeck and Wood, 1977). In this example, we show the growing average 'distance in configuration space', defined as Δr where $\Delta r^2 = |\Delta \mathbf{r}|^2 = (1/N) \sum |\mathbf{r}_i(t) - \mathbf{r}_i^0(t)|^2$, $\mathbf{r}_i^0(t)$ being the position of molecule i at time t in a reference simulation, and $\mathbf{r}_i(t)$ being the position of the same molecule at the same time in the perturbed simulation. In the three cases illustrated here, all the molecules in the perturbed runs are initially displaced in random directions from their reference positions at $t = 0$, by $10^{-3}\sigma$, $10^{-6}\sigma$, and $10^{-9}\sigma$ respectively, where σ is the molecular diameter. In all other respects, the runs are identical; in particular, each corresponds

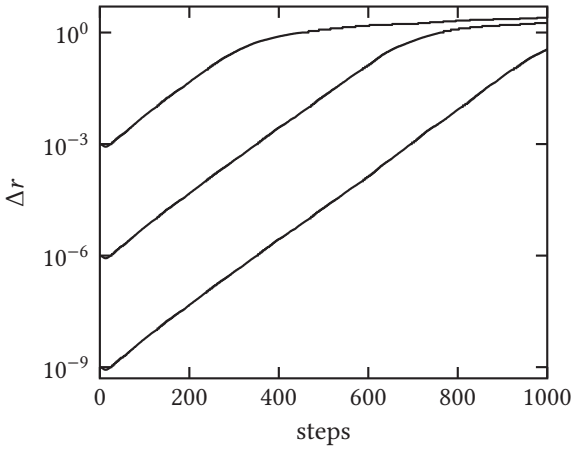

Fig. 3.1 The divergence of trajectories in molecular dynamics. Atoms interacting through the potential $v^{\mathrm{RLJ}}(r)$, eqn (1.10a), were used, and a dense fluid state was simulated ($\rho^* = 0.6$, $T^* = 1.05$, $\delta t^* = 0.005$). Δr is the phase space separation between perturbed and reference trajectories. These simulations used the velocity Verlet algorithm, eqn (3.11), but the results are essentially determined by the equations of motion rather than the integration algorithm.

to essentially the same total energy. As the runs proceed, however, other mechanical quantities eventually become statistically uncorrelated. Typically, properties such as the kinetic energy or pressure remain very close for a period whose length depends on the size of the initial perturbation; after this point the differences become noticeable very rapidly. Presumably, both the reference trajectory and the perturbed trajectory are diverging from the true solution of Newton's equations.

Clearly, no integration algorithm will provide an essentially exact solution for a very long time. Fortunately, we do not need to do this. Remember that molecular dynamics serves two roles. First, we need essentially exact solutions of the equations of motion for times comparable with the correlation times of interest so that we may accurately calculate time correlation functions. Second, we use the method to generate states sampled from the microcanonical ensemble. We do not need exact classical trajectories to do this but must lay great emphasis on energy conservation as being of primary importance for this reason. Momentum conservation is also important, but this can usually be easily arranged. The point is that the particle trajectories must stay on the appropriate constant-energy hypersurface in phase space, otherwise correct ensemble averages will not be generated. Energy conservation is degraded as the timestep is increased, and so all simulations involve a trade-off between economy and accuracy: a good algorithm permits a large timestep to be used while preserving acceptable energy conservation. Other factors dictating the energy-conserving properties are the shape of the potential-energy curves and the typical particle velocities. Thus, shorter timesteps are used at high temperatures, for light molecules and for rapidly varying potential functions.

The final quality an integration algorithm should possess is simplicity. A simple algorithm will involve the storage of only a few coordinates, velocities, etc., and will be

Code 3.1 Velocity Verlet algorithm

This snippet shows a direct translation from the so-called split-operator form of the algorithm (see Section 3.2.2). We have inserted a reminder that the arrays `r`, `v`, and `f`, are dimensioned to contain the entire set of $3N$ positions, velocities, and accelerations, and so the assignment statements apply to the entire array in each case. The (optional) syntax `r(:,:)` emphasizes this, but here, and henceforth, we omit it for brevity.

```
REAL, DIMENSION(3,n) :: r, v, a
v = v + 0.5 * dt * a
r = r + dt * v
! ... evaluate forces and hence a=f/m from r
v = v + 0.5 * dt * a
```

easy to program. Bearing in mind that solution of ordinary differential equations is a fairly routine task, there is little point in wasting valuable man-hours on programming a very complicated algorithm when the time might be better spent checking and optimizing the calculation of forces (see Chapter 5). Little computer time is to be gained by increases in algorithm speed, and the consequences of making a mistake in coding a complicated scheme might be significant.

For all these reasons, most molecular dynamics programs use a variant of the algorithm initially adopted by Verlet (1967) and attributed to Störmer (Gear, 1971). We describe this method in the following section.

3.2.1 The Verlet algorithm

Perhaps the most revealing way of writing the Verlet algorithm is in the so-called velocity Verlet form (Swope et al., 1982), which acts over a single timestep from t to $t + \delta t$ as follows:

$$\mathbf{v}(t + \tfrac{1}{2}\delta t) = \mathbf{v}(t) + \tfrac{1}{2}\delta t\, \mathbf{a}(t) \tag{3.11a}$$

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \delta t\, \mathbf{v}(t + \tfrac{1}{2}\delta t) \tag{3.11b}$$

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t + \tfrac{1}{2}\delta t) + \tfrac{1}{2}\delta t\, \mathbf{a}(t + \delta t). \tag{3.11c}$$

The first step (3.11a) can be thought of as 'half-advancing' the velocities $\mathbf{v}$ to an intermediate time $t + \frac{1}{2}\delta t$, using the values of the accelerations $\mathbf{a}$ at time t; these mid-step velocities are then used to propel the coordinates from time t to $t + \delta t$ in step (3.11b). After this, a force evaluation is carried out to give $\mathbf{a}(t + \delta t)$ for the last step (3.11c) which completes the evolution of the velocities. The equations translate almost directly into computer code, as shown in Code 3.1. At the end of the step, we can calculate quantities such as the kinetic energy by summing the squares of the velocities, or the total momentum vector, by summing the different Cartesian components of the velocity. The potential energy at time $t + \delta t$ will have been computed in the force loop.

This method is numerically stable, convenient, and simple. It is exactly reversible in time and, given conservative forces, is guaranteed to conserve linear momentum. The

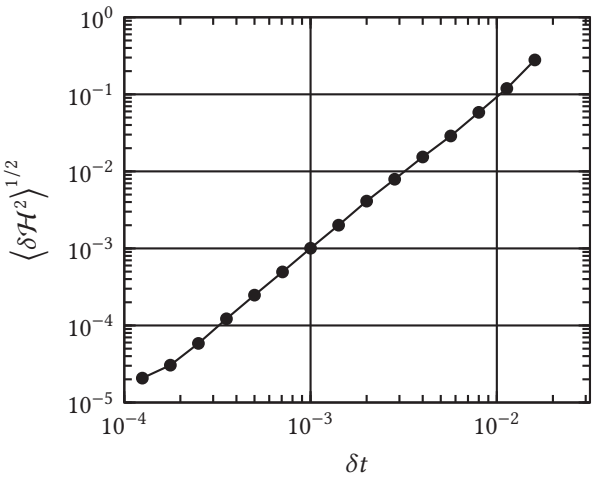

Fig. 3.2 Energy conservation of the Verlet algorithm. The system studied is as for Fig. 3.1. We calculate RMS energy fluctuations $\langle \delta\mathcal{H}^2 \rangle^{1/2}$ for various runs starting from the same initial conditions, and proceeding for the same total simulation time t_{run}, but using different timesteps δt and corresponding numbers of steps $\tau_{\text{run}} = t_{\text{run}}/\delta t$. The plot uses log–log scales.

method has been shown to have excellent energy-conserving properties even with long timesteps. As an example, for simulations of liquid argon near the triple point, RMS energy fluctuations $\langle \delta\mathcal{H}^2 \rangle^{1/2}$ of the order 0.01 % of the potential well depth are observed using $\delta t \approx 10^{-14}$ s, and these increase to 0.2 % for $\delta t \approx 4 \times 10^{-14}$ s (Verlet, 1967; Fincham and Heyes, 1982; Heyes and Singer, 1982). In fact, $\langle \delta\mathcal{H}^2 \rangle^{1/2}$ is closely proportional to δt^2 for Verlet-equivalent algorithms, as shown in Fig. 3.2. As we shall see in the next section, there is an interesting theoretical derivation of this version of the algorithm, which clarifies the reason for this dependence.

In the original 'velocity Verlet' paper (Swope et al., 1982), the previous equations were written in the slightly different form

$$\mathbf{r}(t + \delta t) = \mathbf{r}(t) + \delta t \mathbf{v}(t) + \tfrac{1}{2}\delta t^2 \mathbf{a}(t) \tag{3.12a}$$

$$\mathbf{v}(t + \delta t) = \mathbf{v}(t) + \tfrac{1}{2}\delta t \Big[\mathbf{a}(t) + \mathbf{a}(t + \delta t) \Big]. \tag{3.12b}$$

These are easily obtained from eqns (3.11) by eliminating the mid-step velocity. However, in practice, the velocity is still incremented in two steps, as the alternative is to (needlessly) store accelerations at both the start and end of the step. This is shown in Code 3.2.

Two other versions of the Verlet algorithm are worth mentioning at this point. The original implementation (Verlet, 1967) makes no direct use of the velocities at all but instead is directly related to the second-order equations (3.3). Consider addition of the equations obtained by Taylor expansion about $\mathbf{r}(t)$

$$\begin{aligned} \mathbf{r}(t + \delta t) &= \mathbf{r}(t) + \delta t \mathbf{v}(t) + \tfrac{1}{2}\delta t^2 \mathbf{a}(t) + \cdots \\ \mathbf{r}(t - \delta t) &= \mathbf{r}(t) - \delta t \mathbf{v}(t) + \tfrac{1}{2}\delta t^2 \mathbf{a}(t) - \cdots \end{aligned} \tag{3.13}$$

Code 3.2 Velocity Verlet algorithm (original)

Here `dt_sq` stores the value of δt^2. The algorithm is equivalent to that of Code 3.1, differing only in that the positions are updated before the mid-step velocities are calculated.

```
r = r + dt * v + 0.5 * dt_sq * a
v = v + 0.5 * dt * a
! ... evaluate forces and hence a=f/m from r
v = v + 0.5 * dt * a
```

to give

$$\mathbf{r}(t+\delta t) = 2\mathbf{r}(t) - \mathbf{r}(t-\delta t) + \delta t^2 \mathbf{a}(t). \tag{3.14}$$

The method is based on positions $\mathbf{r}(t)$, accelerations $\mathbf{a}(t)$, and the positions $\mathbf{r}(t-\delta t)$ from the previous step. The Verlet algorithm is 'centered' (i.e. $\mathbf{r}(t-\delta t)$ and $\mathbf{r}(t+\delta t)$ play symmetrical roles in eqn (3.14)), making it time-reversible. It is straightforward to show that eqn (3.14) is equivalent to eqns (3.11), by considering two successive steps and eliminating the velocities.

The velocities are not needed to compute the trajectories, but they are useful for estimating the kinetic energy (and hence the total energy), as well as other interesting properties of the system. They may be obtained from the formula

$$\mathbf{v}(t) = \frac{\mathbf{r}(t+\delta t) - \mathbf{r}(t-\delta t)}{2\delta t}. \tag{3.15}$$

Whereas eqn (3.14) is correct except for errors of order δt^4 (the local error) the velocities from eqn (3.15) are subject to errors of order δt^2. More accurate estimates of $\mathbf{v}(t)$ can be made if more variables are stored, but this adds to the inconvenience already implicit in eqn (3.15), namely that $\mathbf{v}(t)$ can only be computed once $\mathbf{r}(t+\delta t)$ is known.

One implementation of the 'classic' Verlet algorithm is indicated in Code 3.3. It should be clear that the 'classic' Verlet algorithm has identical stability properties to the velocity form, and is very simple. Against it, we must say that the handling of velocities is rather awkward and that the form of the algorithm may needlessly introduce some numerical imprecision (Dahlquist and Björk, 1974). This arises because, in eqn (3.14), a small term ($O(\delta t^2)$) is added to a difference of large terms ($O(\delta t^0)$), in order to generate the trajectory.

Another alternative is the so-called half-step 'leapfrog' scheme (Hockney, 1970; Potter, 1972, Chapter 5). The origin of the name becomes apparent when we write the algorithm down:

$$\mathbf{v}(t+\tfrac{1}{2}\delta t) = \mathbf{v}(t-\tfrac{1}{2}\delta t) + \delta t \mathbf{a}(t) \tag{3.16a}$$

$$\mathbf{r}(t+\delta t) = \mathbf{r}(t) + \delta t \mathbf{v}(t+\tfrac{1}{2}\delta t). \tag{3.16b}$$

The stored quantities are the current positions $\mathbf{r}(t)$ and accelerations $\mathbf{a}(t)$ together with the mid-step velocities $\mathbf{v}(t-\frac{1}{2}\delta t)$. The velocity equation eqn (3.16a) is implemented first,

Code 3.3 Classic Verlet algorithm

The 'classic' Verlet algorithm evaluates accelerations from the current positions, then uses these together with the old positions in the advancement step. The variable `dt_sq` stores δt^2 as usual. During this step, it is possible to calculate the current velocities. We handle this using a temporary array `r_new` to store the new positions. Then, a shuffling operation takes place in the last two statements. At the end of the step, the positions have been advanced, but the 'current' (now 'old') potential energy can be combined with the kinetic energy, calculated from the 'current' velocities. Following the particle move, we are ready to evaluate the forces at the start of the next step.

```
REAL, DIMENSION(3,n) :: r, r_old, r_new, a, v
! ... evaluate forces and hence a=f/m from r
r_new  = 2.0 * r - r_old + dt_sq * a
v      = ( r_new - r_old ) / ( 2.0 * dt )
r_old  = r
r      = r_new
```

and the velocities leap over the coordinates to give the next mid-step values $\mathbf{v}(t + \frac{1}{2}\delta t)$. During this step, the current velocities may be calculated

$$\mathbf{v}(t) = \tfrac{1}{2}\Big(\mathbf{v}(t + \tfrac{1}{2}\delta t) + \mathbf{v}(t - \tfrac{1}{2}\delta t)\Big). \tag{3.17}$$

This is necessary so that the energy ($\mathcal{H} = \mathcal{V} + \mathcal{K}$) at time t can be calculated, as well as any other quantities that require positions and velocities at the same instant. Following this, eqn (3.16b) is used to propel the positions once more ahead of the velocities. After this, the new accelerations may be evaluated ready for the next step.

Elimination of the velocities from these equations shows that the method is algebraically equivalent to Verlet's algorithm. In fact, eqn (3.16b) is identical to eqn (3.11b), while eqn (3.16a) is obtained by combining (3.11a) with (3.11c) *for the previous step*. Numerical benefits derive from the fact that at no stage do we take the difference of two large quantities to obtain a small one; this minimizes loss of precision on a computer. As is clear from eqn (3.17), the leapfrog method still does not handle the velocities in a completely satisfactory manner, and velocity Verlet is generally preferable. A complete molecular dynamics program for Lennard-Jones atoms, using the velocity Verlet method, is given in Code 3.4.

3.2.2 Formal basis of Verlet algorithm

To an extent, there is no need to understand 'where algorithms come from', as long as they work. Nonetheless, an understanding of the formal background to molecular dynamics algorithms, and particularly the Verlet algorithm, has been extremely useful in terms of knowing their limitations and how they may be extended to different situations. We shall only scratch the surface: for more details the reader is referred to the books of Leimkuhler and Reich (2004) and Tuckerman (2010). The following equations make use of the Liouville operator, introduced in eqn (2.4), and its exponential $\exp(\mathrm{i}Lt) \equiv U(t)$, which

Code 3.4 Molecular dynamics, NVE-ensemble, Lennard-Jones

These files are provided online. The program `md_nve_lj.f90` controls the simulation, reads in the run parameters, implements the velocity Verlet algorithm, and writes out the results. It uses the routines in `md_lj_module.f90` to evaluate the Lennard-Jones forces, and various utility modules (see Appendix A) for input/output and simulation averages. Code to set up an initial configuration is provided in `initialize.f90`.

```
! md_nve_lj.f90
! Molecular dynamics, NVE ensemble
PROGRAM md_nve_lj

! md_lj_module.f90
! Force routine for MD simulation, Lennard-Jones atoms
MODULE md_module
```

is often called the *propagator*: it has the effect of moving the system (i.e. the coordinates, momenta, and all the dynamical variables that depend on them) forward through time.

In the Verlet algorithm we use an approximate form of the propagator, which arises from splitting iL in two (Tuckerman et al., 1992):

$$\mathrm{i}L = \dot{\mathbf{r}} \cdot \frac{\partial}{\partial \mathbf{r}} + \dot{\mathbf{p}} \cdot \frac{\partial}{\partial \mathbf{p}} = \mathbf{v} \cdot \frac{\partial}{\partial \mathbf{r}} + \mathbf{f} \cdot \frac{\partial}{\partial \mathbf{p}} \equiv \mathrm{i}L_1 + \mathrm{i}L_2, \tag{3.18}$$

where, as before, we abbreviate $\mathbf{r}$, $\mathbf{v}$ for the complete set of positions, velocities, etc. It is not hard to see that the 'propagators' corresponding to each of the separate parts will only affect the corresponding coordinate

$$\exp(\mathrm{i}L_1 \delta t)\, \mathbf{r} = \mathbf{r} + \mathbf{v}\delta t \qquad \exp(\mathrm{i}L_1 \delta t)\, \mathbf{p} = \mathbf{p} \qquad \text{drift,} \tag{3.19}$$

$$\exp(\mathrm{i}L_2 \delta t)\, \mathbf{r} = \mathbf{r} \qquad \exp(\mathrm{i}L_2 \delta t)\, \mathbf{p} = \mathbf{p} + \mathbf{f}\delta t \qquad \text{kick.} \tag{3.20}$$

The first of these is termed the 'drift' because it advances coordinates without changing momenta, rather like drifting in free flight, with the forces switched off. The second is called the 'kick' since it impulsively changes momenta without altering positions. It is important to realize that each of these separate propagation steps has been derived from a corresponding part of the Hamiltonian: the 'drift' arises from the kinetic-energy part, while the 'kick' comes from the potential-energy part.

Now, much like operators in quantum mechanics, iL_1 and iL_2 do not commute with each other, and this means that the following relation

$$\exp(\mathrm{i}L\delta t) = \exp[(\mathrm{i}L_1 + \mathrm{i}L_2)\delta t] \approx \exp(\mathrm{i}L_1 \delta t) \exp(\mathrm{i}L_2 \delta t)$$

is only an approximation, not an exact relation. The error associated with the approximation is, however, 'small', that is, it becomes asymptotically exact in the limit $\delta t \to 0$. A slightly different approximation would result from combining the two partial propagators

in the opposite order, and the following arrangement has the additional merit of being exactly time-reversible:

$$\exp(\mathrm{i}L\delta t) \approx \exp(\mathrm{i}L_2\delta t/2)\ \exp(\mathrm{i}L_1\delta t)\ \exp(\mathrm{i}L_2\delta t/2). \tag{3.21}$$

The operators act in turn, reading from right to left, upon the phase space variables $\mathbf{r}$ and $\mathbf{p}$, initially at time t, converting them into the new variables at $t + \delta t$. An attractive feature of this formalism is that the three successive steps embodied in eqn (3.21), with the operators defined by eqns (3.19) and (3.20), translate directly (Martyna et al., 1996) into the velocity Verlet algorithm of eqn (3.11):

(a) 'half-kick', $\mathbf{r}$ constant, $\mathbf{p}(t) \to \mathbf{p}(t + \frac{1}{2}\delta t) = \mathbf{p}(t) + \frac{1}{2}\delta t\, \mathbf{f}(t)$;
(b) 'drift', free flight with $\mathbf{p}$ constant, $\mathbf{r}(t) \to \mathbf{r}(t + \delta t) = \mathbf{r}(t) + \delta t\, \mathbf{p}(t + \frac{1}{2}\delta t)/m$;
(c) 'half-kick', $\mathbf{r}$ constant, $\mathbf{p}(t + \frac{1}{2}\delta t) \to \mathbf{p}(t + \delta t) = \mathbf{p}(t + \frac{1}{2}\delta t) + \frac{1}{2}\delta t\, \mathbf{f}(t + \delta t)$.

This particular splitting is quite simple; possible advantages of a higher-order decomposition have been discussed by Ishida et al. (1998).

A key consequence of the propagators when split in this way (the so-called symplectic property) is that, although the trajectories are approximate and will not conserve the true energy $\mathcal{H}$, they do exactly conserve a 'shadow Hamiltonian' $\mathcal{H}^{\ddagger}$ (Toxvaerd, 1994), where $\mathcal{H}$ and $\mathcal{H}^{\ddagger}$ differ from each other by a small amount, vanishing as $\delta t \to 0$. More precisely, it may be shown that the difference $\mathcal{H} - \mathcal{H}^{\ddagger}$ can be written as a Taylor expansion in δt, where the coefficients involve derivatives of $\mathcal{H}$ with respect to the coordinates. The consequence is that the system will remain on a hypersurface in phase space which is 'close' to the true constant-energy hypersurface. Such a stability property is extremely useful in molecular dynamics, since we wish to sample constant-energy states. It essentially eliminates any long-term 'drift' in the total energy.

We can illustrate this with the example of the simple one-dimensional harmonic oscillator for which the trajectory generated by the velocity Verlet algorithm and the corresponding shadow Hamiltonian may be written down explicitly (Venneri and Hoover, 1987; Toxvaerd, 1994). For natural frequency ω, the exact equations of motion and conserved Hamiltonian are

$$\dot{r} = p/m, \quad \dot{p} = -m\omega^2 r, \quad \mathcal{H}(r,p) = p^2/2m + \tfrac{1}{2}m\omega^2 r^2. \tag{3.22}$$

The shadow Hamiltonian depends on timestep through a quantity $\zeta = 1 - (\omega\delta t/2)^2$ and may be written

$$\mathcal{H}^{\ddagger}(r,p) = p^2/2m + \tfrac{1}{2}m\omega^2\zeta r^2. \tag{3.23}$$

It would be equally valid to divide the right-hand side by the factor ζ, in which case the timestep dependence would be associated with the kinetic-energy term, or by $\sqrt{\zeta}$, when it would appear in both terms; for all these choices the difference is $O((\omega\delta t)^2)$. The present choice allows us to compare trajectories which initially coincide at $r = 0$ and have the same momentum and energy, and we do this in the (r,p) 'phase portraits' of Fig. 3.3. The true dynamics follows an elliptical trajectory defined by $\mathcal{H}$ = constant (in the figure this is a circle). The equation $\mathcal{H}^{\ddagger}$ = constant also describes an ellipse, differing only slightly (for small $\omega\delta t$) from the true one. On this diagram, the 'kicks' are vertical line segments, and the 'drifts' are horizontal ones. At the end of each velocity Verlet step the discrete

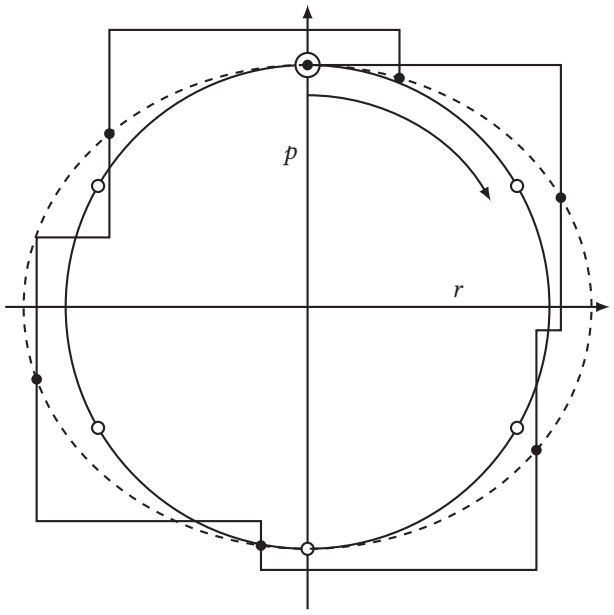

Fig. 3.3 Exact trajectory (circle) and velocity Verlet trajectory (straight line segments) for the harmonic oscillator. The shadow-Hamiltonian conservation law is shown by a dashed ellipse. The open and closed circles mark out phase points at the end of each timestep for the corresponding trajectories. The starting point is marked with a double circle. The timestep is chosen such that $\omega\delta t = \pi/3$, so the exact trajectory returns to its starting point after six timesteps.

trajectory lands exactly on the constant-$\mathcal{H}^\ddagger$ ellipse, although the intermediate stages lie off the ellipse. Therefore, the long-time trajectory is very stable in this sense: it will never leave the ellipse $\mathcal{H}^\ddagger \approx \mathcal{H} + O((\omega\delta t)^2) = \text{constant}$. However, it can also be seen that the positions and coordinates at the end of each timestep are quickly losing their phase, relative to the corresponding points on exact trajectory. Of course, the example shown uses a very large timestep, to emphasize all these effects.

3.3 Molecular dynamics of rigid non-spherical bodies

Molecular systems are not rigid bodies in any sense: they consist of atoms interacting via intra- and inter-molecular forces. In principle, we should not distinguish between these forces, but as a practical definition we take the forces acting within molecules to be at least an order of magnitude greater than those acting between molecules. If treated classically, as in the earliest molecular simulations (Harp and Berne, 1968; 1970; Berne and Harp, 1970), molecular bond vibrations would occur so rapidly that an extremely short timestep would be required to solve the equations of motion. We return to this approach in Section 3.5; however, we must bear in mind that the classical approach is highly questionable for bond vibrations. A common solution to these problems is to take the intramolecular bonds to be of fixed length. This is not an inconsequential step, but seems reasonable if, as is commonly true at normal temperatures, the amplitude