require sums over all $\frac{1}{2}N(N-1)$ pairs of particles. If the interactions are short range, the time required for these sums can be reduced to approximately order N. The idea is to take advantage of the fact that many pairs of particles are separated by a distance much greater than the effective range r_c of the interparticle interaction. For example, if the distance between two particles interacting via the Lennard–Jones potential is sufficiently large, the magnitude of the potential is so small that it can be neglected. Popular choices for the cutoff r_c are 2.3σ and 2.5σ. The use of a cutoff is equivalent to assuming that $u(r)$ in (8.2) is given by the usual Lennard–Jones form for $r < r_c$ and is zero for $r > r_c$. However, this use of a cutoff implies that $u(r)$ has a discontinuity at $r = r_c$, which means that whenever a particle pair "crosses" the cutoff distance, the energy jumps, thus affecting the apparent energy conservation. To avoid this problem, it is a good idea to modify the potential so as to eliminate the discontinuity in both $u(r)$ and the force $-du/dr$. Hence, we write

$$\tilde{u}(r) = u(r) - u(r_c) - \left.\frac{du(r)}{dr}\right|_{r=r_c} (r - r_c), \tag{8.42}$$

where $u(r)$ is the usual Lennard–Jones potential.

The use of the interparticle potential (8.42) to calculate the force and the energy requires considering only those pairs of particles whose separation is less than r_c. Because testing whether each pair satisfies this criterion is an order N^2 calculation, we have to limit the number of pairs tested. One way is to divide the box into small cells and to only compute the distance between particles that are in the same cell or in nearby cells. Another method is to maintain a list for each particle of its neighbors whose separation is less than a distance r_n, where r_n is chosen to be slightly greater than r_c. The idea is to use the same list of neighbors for several time steps (usually 10–20) so that the time consuming job of updating the list of neighbors does not have to be done too often. The cell method and the neighbor list method do not become efficient until N is approximately a few hundred particles.

Usually, the neighbor list leads to the consideration of fewer particle pairs in the force calculation than the cell list. We provide a method to compute the neighbor list below. A more efficient approach is to use cells to construct the neighbor list.

```
public void computeNeighborList() {
   for(int i = 0; i < N-1; i++) {
      numberInList[i] = 0;
      for(int j = i+1; j < N; j++) {
         double dx = separation(x[i] - x[j],Lx);
         double dy = separation(y[i] - y[j],Ly);
         double r2 = dx*dx + dy*dy;
         if(r2 < r2ListCutoff) {
            list[i][numberInList[i]] = j;
            numberInList[i]++;
         }
      }
   }
}
```

To use this list in method `computeAcceleration`, we replace the for loops by

```
for (int i = 0; i < N-1; i++) {
   for (int k = 0; k < numberInList[i]; k++) {
      int j = list[i][k];
   }
}
```

The method `computeNeighborList` should be called before a particle may have moved a distance equal to the difference $r_n - r_c$. This time depends on the density and the temperature. For dense systems a reasonable value for r_n is 2.7σ. Simulations of small systems can be used to determine the time between calls of `computeNeighborList`.

Note that in method `computeNeighborList`, only particles $j > i$ are included in `list[i]`. In Section 15.10 we will consider Monte Carlo simulations where a particle is chosen at random, and its potential energy of interaction must be computed. In this case we cannot take advantage of Newton's third law, and a neighbor list must be created for all particles that are within a distance r_n of particle i.

*Problem 8.21 Neighbor lists

(a) Simulate a system of $N = 64$ Lennard–Jones particles in a square cell with $L = 10$ at a temperature $T = 2.0$. After the system has reached equilibrium, determine the shortest time for any particle to move a distance equal to 0.2. Use half this time in the rest of the program as the time between updates of the neighbor list.

(b) Run your simulation with and without the neighbor list starting from identical initial configurations. Choose $r_c = 2.3$ and use the modified potential given in (8.42). Calculate $g(r)$, the pressure, the heat capacity (see Problem 8.8), and the temperature. Make sure your results are identical. Compare the amount of CPU time with and without the use of the neighbor list.

(c) Repeat part (b) with $N = 256$ but with the same density and total energy. You can adjust the total energy by scaling the initial velocities. Increase N until the CPU time for the neighbor list version is faster.

(d) Continue increasing the number of particles by a factor of four, but only use the program with the neighbor list. Determine the CPU time required for one time step as a function of N. ■

So far we have discussed molecular dynamics simulations at fixed energy, volume, and number of particles. Molecular dynamics simulations at fixed temperature are discussed in Project 8.24. It is also possible to modify the dynamics so as to do molecular dynamics simulations at constant pressure and to do simulations in which the shape of the cell is determined by the dynamics, rather than imposed by the program. Such a simulation is essential for the study of solid-to-solid transitions where the major change is the shape of the crystal.

In addition to these algorithmic advances, there is much more to learn about the properties of the system. For example, how are transport properties such as the viscosity and the thermal conductivity related to the trajectories? We have also not discussed one of the most fundamental properties of a many-body system, namely, its entropy. In brief, not all macroscopic properties of a many-body system, including the entropy, can be defined as a time average over some function of the phase space coordinates of the particles (but see Ma). However, changes in the entropy can be computed by using thermodynamic integration.

The fundamental limitation of molecular dynamics is the existence of *multiple time scales*. We must choose the time step Δt to be smaller than any physical time scale in the system. For a solid, the smallest time scale is the period of the oscillatory motion of individual particles about their equilibrium positions. If we want to know how the solid responds to the addition of an interstitial particle or a vacancy, we would have to run the simulation for millions of small time steps for the vacancy to move several interparticle