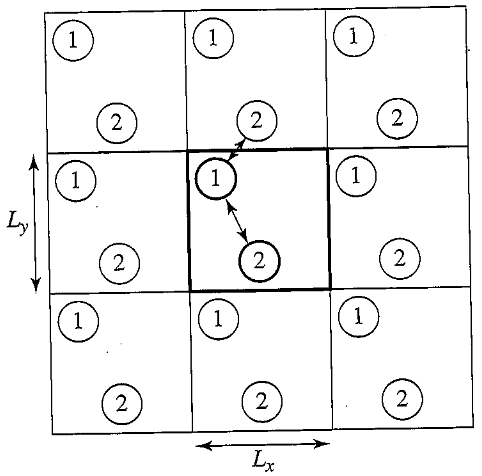

Figure 8.3 Example of the minimum image approximation in two dimensions. The minimum image distance convention implies that the separation between particles 1 and 2 is given by the lesser of the two distances shown.

(b) From the results of part (a) we might consider writing `x = x % L` as an alternative to Listing 8.1. What about negative values of x? In this case `-17 % 5 = -2`. Because we want the resultant position to be positive, we write

```
return x<0 ? x\%L+L: x\%L;
```

Explain this syntax and write a program to test if this statement works as claimed.

(c) Write a simple program to determine if the `%` operator is faster than the if-else construction in Listing 8.1. Write another program that compares the speed of calling the `PCB.position` method to that of inlining the PBC code. In other words, replace the method call by the above statement. ■

We now discuss the nature of periodic boundary conditions. Imagine a set of N particles in a two-dimensional box or cell. The use of periodic boundary conditions implies that the central cell is duplicated an infinite number of times to fill the space. Figure 8.3 shows the first several image cells for $N = 2$. The shape of the central cell must be such that the cell fills space under successive translations. Each image cell contains the original particles in the same relative positions as the central cell. That is, periodic boundary conditions yield an infinite system, although the positions of the particles in the image cells are identical to the positions of the particles in the central cell. These boundary conditions also imply that every point in the cell is equivalent and that there is no surface.

As a particle moves in the original cell, its periodic images move in the image cells. Hence only the motion of the particles in the central cell needs to be followed. When a particle enters or leaves the central cell, the move is accompanied by an image of that particle leaving or entering a neighboring cell through the opposite face.

The total force on a given particle i is due to the force from every other particle j within the central cell and from the periodic images of particle j. That is, if particle i interacts with particle j in the central cell, then particle i interacts with *all* the periodic replicas of particle j. Hence, in general, there are an infinite number of contributions to the force on any given particle. For long-range interactions such as the Coulomb potential, these contributions have to be included using special methods. For short-range interactions, we can reduce the number of contributions by adopting the minimum image approximation, which assumes that particle i in the central cell interacts only with the nearest image of particle j; the interaction is set equal to zero if the distance of the image from particle i is greater than $L/2$. An example of the minimum image approximation is shown in Figure 8.3.

8.6 ■ A MOLECULAR DYNAMICS PROGRAM

In this section we develop a molecular dynamics program to simulate a two-dimensional system of particles interacting via the Lennard–Jones potential. We choose two rather than three dimensions because it is easier to visualize the results and the calculations are not as time consuming.

In principle, we could define a class for a particle and instantiate an object for each particle. However, this use would be very inefficient and would take up more memory and CPU time than using one class to represent all N particles. Instead we will store the x- and y-components of the positions and velocities in the `state` array and store the accelerations of the particles in a separate array. As usual, we will develop two classes, `LJParticles` and `LJParticlesApp`.

Because the system is deterministic, the nature of the motion is determined by the initial conditions. An appropriate choice of the initial conditions is more difficult than might first appear. For example, how can we choose the initial configuration (a set of positions and velocities) to correspond to a liquid at a desired temperature? According to the equipartition theorem, the mean kinetic energy of a particle per degree of freedom is $kT/2$, where k is Boltzmann's constant and T is the temperature. We can generalize this relation to define the temperature at time t by

$$kT(t) = \frac{2}{d}\frac{K(t)}{N} = \frac{1}{Nd}\sum_{i=1}^{N} m_i \mathbf{v}_i(t)\cdot\mathbf{v}_i(t), \tag{8.5}$$

where K is the total kinetic energy of the system, $\mathbf{v}_i$ is the velocity of particle i with mass m_i, and d is the spatial dimension of the system.

We can use (8.5) to choose an initial set of velocities. The following method gives the particles a random set of velocities, sets the total velocity (momentum) to zero, and then rescales the velocities so that the desired initial kinetic energy is achieved.

Listing 8.3 Method for choosing the initial velocities.

```
public void setVelocities() {
   double vxSum = 0.0;
   double vySum = 0.0;
   for(int i = 0;i<N;++i) { // assign random initial velocities
      state[4*i+1] = Math.random() - 0.5;   // vx
      state[4*i+3] = Math.random() - 0.5;   // vy
      vxSum += state[4*i+1];
      vySum += state[4*i+3];
   }
```