time origins, we obtain

$$\overline{R(t=1)^2} = \frac{1}{3}[(x(1)-x(0))^2 + (x(2)-x(1))^2 + (x(3)-x(2))^2]$$
$$= \frac{1}{3}[0.0009 + 0.0484 + 0.1444]v = 0.0646$$
$$\overline{R(t=2)^2} = \frac{1}{2}[(x(2)-x(0))^2 + (x(3)-x(1))^2] = \frac{1}{2}[0.0361 + 0.36] = 0.1981$$
$$\overline{R(t=3)^2} = [x(3)-x(0)]^2 = 0.3249.$$

Note that there are fewer combinations of the positions as the time difference increases.

In Listing 8.21 we show a method that computes $R(t)^2$ assuming that the positions of all N particles have been collected for `n` times in the arrays `xSave[i][k]` and `ySave[i][k]`; the time is indexed by k. Because of periodic boundary conditions, we cannot find the distance moved by a particle by just keeping track of its position. Imagine that a particle moved in only one direction and returned to its original position. If we just subtracted the coordinates of the position, we would find that the particle's displacement was zero when in fact it really moved an amount equal to the length of the simulation cell. To keep track of the movement of each particle, we use two arrays, `xWrap` and `yWrap`. Every time particle i moves past the right boundary in the time interval `k*dk` to `(k+1)*dk`, `xWrap[i][k]` is incremented by `Lx`. Similarly, each time the particle moves past the left boundary `xWrap[i][k]` is decremented by `Lx`. A similar procedure is used for `yWrap`.

Listing 8.21 Listing of method `computeR2` for finding the mean square displacement.

```
public void computeR2(PlotFrame data) {
   for(int dk = 1; dk < n-1; ++dk) { // loops over time intervals
      int norm = 0;
      double r2 = 0;
      for(int i = 0; i < N; i++) { // loops over particles
         // time origin labeled by k0
         // loops over time origins
         for(int k0 = 0; k0 < n-dk-1; ++k0)  {
          double dx = (xSave[i][k0+dk]+xWrap[i][k0+dk])-
               (xSave[i][k0]+xWrap[i][k0+dk]);
          double dy = (ySave[i][k0+dk]+yWrap[i][k0+dk])-
               (ySave[i][k0]+yWrap[i][k0+dk]);
          r2 += dx*dx + dy*dy;
          norm++;
        }
      }
      data.append(0,dk*timeInterval,r2/norm);
   }
}
```

We show our results for $\overline{R(t)^2}$ for a system of Lennard–Jones particles in Figure 8.9. Note that $\overline{R(t)^2}$ increases approximately linearly with t, with a slope of roughly 0.61. From (8.39) the corresponding self-diffusion coefficient is $D = 0.61/4 \approx 0.15$. In Problem 8.19 we use method `computeR2` to compute the self-diffusion coefficient. An easier but less direct way of computing D is discussed in Project 8.23.

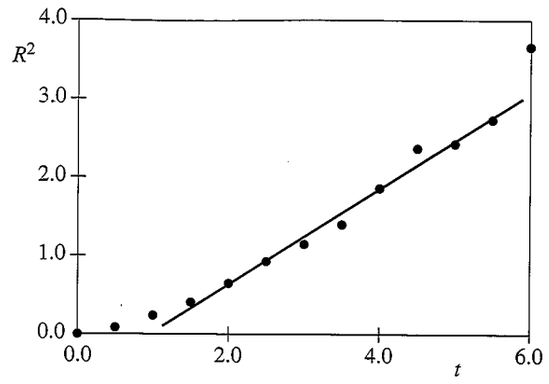

Figure 8.9 The time dependence of the mean square displacement $\overline{R(t)^2}$ for one particle in a two-dimensional Lennard–Jones system with $N = 16$, $L = 5$, and $E = 5.8115$. The position of a particle was saved at intervals of 0.5. Much better results can be obtained by averaging over all particles and over a longer run. The least squares fit was made between $t = 1.5$ and $t = 5.5$. As expected, this fit does not pass through the origin. The slope of the fit is 0.61.

Problem 8.19 The self-diffusion coefficient

(a) Use either your hard disk or molecular dynamics program and visually follow the motion of a particular particle by "tagging" it, for example, by drawing its path with a different color. Describe its motion qualitatively.

(b) Modify your program so that the coordinates of the particles are saved at regular intervals (see Appendix 8A). The optimum time interval needs to be determined empirically. If you save the coordinates too often, the data file will become very large, and you will waste time saving the coordinates. If you do not save the positions often enough, you will lose information. Because the time step Δt must be small compared to any interesting time scale, we know that the time interval for saving the positions must be at least an order of magnitude greater than Δt. A good first guess is to choose the time interval for saving the coordinates to be the order of $10\Delta t$. The easiest procedure for hard disks is to save the coordinates at intervals measured in terms of the number of collisions. If you average over a sufficient number of collisions, you can find the relation between the elapsed time and the number of collisions.

(c) For a finite system, the time difference t cannot be chosen to be too large because the displacement of a particle is bounded. What is the maximum value of $R^2(t)$?

(d) Compute $\overline{R(t)^2}$ for conditions that correspond to a dense fluid. Does $\overline{R(t)^2}$ increase as t^2 as for a free particle or more slowly? Does $\overline{R(t)^2}$ increase linearly with t for longer times?

(e) Use the relation (8.39) to estimate the magnitude of D from the slope of $\overline{R(t)^2}$ for the time interval for which $\overline{R(t)^2}$ is approximately linear. Obtain D for several different temperatures and densities. (A careful study of $\overline{R(t)^2}$ for much larger systems and much longer times would show that $\overline{R(t)^2}$ is not proportional to t in two dimensions. Instead, $\overline{R(t)^2}$ has a term proportional to $t \log t$, which dominates the linear t term