

Chapter 8

The Dynamics of Many Particle Systems

©2000 by Harvey Gould and Jan Tobochnik
6 December 2000

We simulate the dynamical behavior of many particle systems and observe their qualitative features. Some of the basic ideas of equilibrium statistical mechanics and kinetic theory are introduced.

8.1 Introduction

Gases, liquids, and solids are examples of systems that contain many mutually interacting particles. Given our knowledge of the laws of physics at the microscopic level, how can we understand the observed behavior of these systems and more complex systems such as polymers and proteins? As an example, consider two cups of water prepared under similar conditions. Each cup contains approximately 10^{24} molecules which, to a good approximation, move according to the laws of classical physics. Although the intermolecular forces produce a complicated trajectory for each molecule, the observable properties of the water in each cup are indistinguishable and are easy to describe. For example, the temperature of the water in each cup is independent of time even though the positions and velocities of the individual molecules are changing continually.

One way to understand the behavior of a many particle system is to begin from the known intermolecular interactions and do a computer simulation of its dynamics. This approach, known as the **molecular dynamics** method, has been applied to systems of several hundred to a million particles and has given us much insight into the behavior of gases, liquids, and solids.

A knowledge of the trajectories of 10^4 or even 10^{24} particles is not helpful unless we know the right questions to ask. What are the useful parameters needed to describe these systems? What are the essential characteristics and regularities exhibited by many particle systems? From our study of chaotic systems, we might suspect that the only meaningful quantities we can compute

are averages over the trajectories, rather than the trajectories themselves. Questions such as these are addressed by statistical mechanics and many of the ideas of statistical mechanics are discussed in this chapter. However, the only background needed for this chapter is a knowledge of Newton's laws of motion.

8.2 The Intermolecular Potential

The first step is to specify the model system we wish to simulate. For simplicity, we assume that the dynamics can be treated classically and that the molecules are spherical and chemically inert. We also assume that the force between any pair of molecules depends only on the distance between them. In this case the total potential energy U is a sum of two-particle interactions:

$$U = u(r_{12}) + u(r_{13}) + \cdots + u(r_{23}) + \cdots = \sum_{i=1}^{N-1} \sum_{j=i+1}^N u(r_{ij}), \quad (8.1)$$

where $u(r_{ij})$ depends only on the magnitude of the distance \mathbf{r}_{ij} between particles i and j . The pairwise interaction form (8.1) is appropriate for simple liquids such as liquid argon.

In principle, the form of $u(r)$ for electrically neutral molecules can be constructed by a first principles quantum mechanical calculation. Such a calculation is very difficult, and it usually is sufficient to choose a simple phenomenological form for $u(r)$. The most important features of $u(r)$ for simple liquids are a strong repulsion for small r and a weak attraction at large r . The repulsion for small r is a consequence of the Pauli exclusion principle. That is, the electron clouds of two molecules must distort to avoid overlap, causing some of the electrons to be in different quantum states. The net effect is an increase in kinetic energy and an effective repulsive force between the electrons, known as **core repulsion**. The dominant weak attraction at larger r is due to the mutual polarization of each molecule; the resultant attractive force is called the **van der Waals** force.

One of the most common phenomenological forms of $u(r)$ is the Lennard-Jones potential:

$$u(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6 \right]. \quad (8.2)$$

A plot of the Lennard-Jones potential is shown in Figure 8.1. The r^{-12} form of the repulsive part of the interaction has been chosen for convenience only. The Lennard-Jones potential is parameterized by a length σ and an energy ϵ . Note that $u(r) = 0$ at $r = \sigma$, and that $u(r)$ is essentially zero for $r > 3\sigma$. The parameter ϵ is the depth of the potential at the minimum of $u(r)$; the minimum occurs at a separation $r = 2^{1/6}\sigma$. The parameters ϵ and σ of the Lennard-Jones potential which give good agreement with the experimental properties of liquid argon are $\epsilon = 1.65 \times 10^{-21}$ J and $\sigma = 3.4$ Å.

Problem 8.1. Qualitative properties of the Lennard-Jones interaction

Write a short program or use a graphics package to plot the Lennard-Jones potential (8.1) and the magnitude of the corresponding force:

$$\mathbf{f}(r) = -\nabla u(r) = \frac{24\epsilon}{r} \left[2\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6 \right] \hat{\mathbf{r}}. \quad (8.3)$$

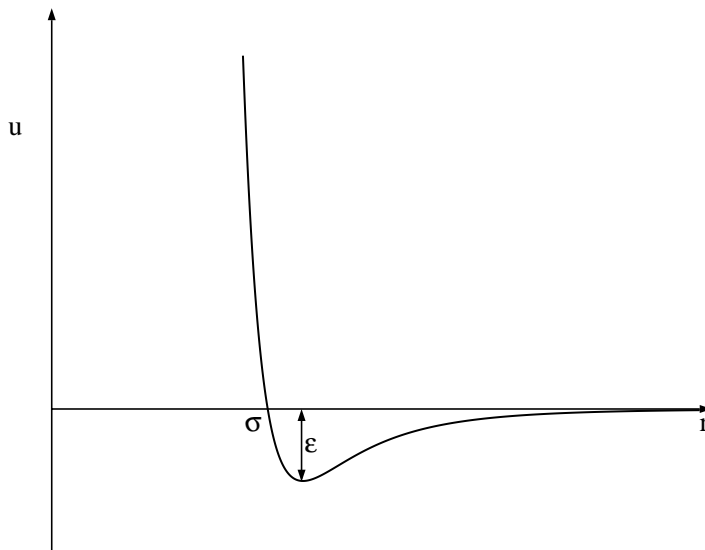


Figure 8.1: Plot of the Lennard-Jones potential $u(r)$. Note that the potential is characterized by a length σ and an energy ϵ .

What is the value of $u(r)$ for $r = 0.8\sigma$? How much does u increase if r is decreased to $r = 0.72\sigma$, a decrease of 10%? What is the value of u at $r = 2.5\sigma$? At what value of r does the force equal zero?

8.3 The Numerical Algorithm

Now that we have specified the interaction between the particles, we need to introduce a numerical integration method for computing the trajectory of each particle. As might be expected, we need to use at least a second-order algorithm to maintain conservation of energy for the times of interest in molecular dynamics simulations. We adopt the commonly used algorithm:

$$x_{n+1} = x_n + v_n \Delta t + \frac{1}{2} a_n (\Delta t)^2 \quad (8.4a)$$

$$v_{n+1} = v_n + \frac{1}{2} (a_{n+1} + a_n) \Delta t. \quad (8.4b)$$

To simplify the notation, we have written the algorithm for only one component of the particle's motion. The new position is used to find the new acceleration a_{n+1} which is used together with a_n to obtain the new velocity v_{n+1} . The algorithm represented by (8.4) is a convenient form of the Verlet algorithm (see [Appendix 5A](#)).

8.4 Boundary Conditions

A useful simulation must incorporate all the relevant features of the physical system of interest. The ultimate goal of our simulations is to understand the behavior of bulk systems—systems of the order of $N \sim 10^{23} - 10^{25}$ particles. In bulk systems the fraction of particles near the walls of the container is negligibly small. However, the number of particles that can be studied in a molecular dynamics simulation is typically $10^3 - 10^5$, although as many as 10^6 particles or more, can be studied on present-day supercomputers. For these small systems the fraction of particles near the walls of the container is significant, and hence the behavior of such a system would be dominated by surface effects.

The most common way of minimizing surface effects and to simulate more closely the properties of a bulk system is to use what are known as **periodic boundary conditions**. First consider a one-dimensional “box” of N particles that are constrained to move on a line of length L . The ends of the line serve as imaginary walls. The usual application of periodic boundary conditions is equivalent to considering the line to be a circle (see Figure 8.2). The distance between the particles is measured along the arc, and hence the maximum separation between any two particles is $L/2$.

The computer code for periodic boundary conditions is straightforward. If a particle leaves the box by crossing a boundary, we add or subtract L to the coordinate. One simple way is to use an IF statement after the particles have been moved:

```
IF x > L then
  LET x = x - L
ELSE IF x < 0 then
  LET x = x + L
END IF
```

To compute the minimum distance dx between particles 1 and 2 at $x(1)$ and $x(2)$ respectively, we can write

```
LET dx = x(1) - x(2)
IF dx > 0.5*L then
  LET dx = dx - L
ELSE IF dx < -0.5*L then
  LET dx = dx + L
END IF
```

The generalization of this application of periodic boundary conditions to two dimensions is straightforward if we imagine a box with opposite edges joined so that the box becomes the surface of a torus (the shape of a doughnut and a bagel).

We now discuss the motivation for this choice of boundary conditions. Imagine a set of N particles in a two-dimensional cell. The use of periodic boundary conditions implies that this central cell is duplicated an infinite number of times to fill two-dimensional space. Each image cell contains the original particles in the same relative positions as the central cell. Figure 8.3 shows the first several image cells for $N = 2$ particles. Periodic boundary conditions yield an infinite system, although the motion of particles in the image cells is identical to the motion of the particles

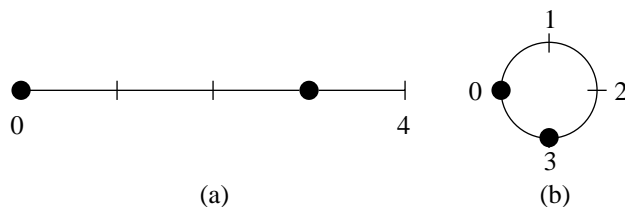


Figure 8.2: (a) Two particles at $x = 0$ and $x = 3$ on a line of length $L = 4$; the distance between the particles is 3. (b) The application of periodic boundary conditions for short range interactions is equivalent to thinking of the line as forming a circle of circumference L . In this case the minimum distance between the two particles is 1.

in the central cell. These boundary conditions also imply that every point in the cell is equivalent and that there is no surface. The shape of the central cell must be such that the cell fills space under successive translations.

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. However, for short range interactions, we may reduce the number of contributions by adopting the **minimum image** or nearest image approximation. This approximation implies 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 condition is shown in Figure 8.3. Note that the minimum image approximation implies that the calculation of the total force on all N particles due to pairwise interactions involves a maximum of $N(N - 1)/2$ contributions.

8.5 Units

To reduce the possibility of roundoff error, it is useful to choose units so that the computed quantities are neither too small nor too large. Because the values of the distance and the energy associated with typical liquids are very small in SI units, we choose the Lennard-Jones parameters σ and ϵ to be the units of distance and energy, respectively. (The values of σ and ϵ for argon are given in Table 8.1.) We also choose the unit of mass to be the mass of one atom, m . We can express all other quantities in terms of σ , ϵ , and m . For example, we measure velocities in units of $(\epsilon/m)^{1/2}$, and the time in units of $\sigma(m/\epsilon)^{1/2}$. If we take $m = 6.69 \times 10^{-26}$ kg, the mass of an argon atom, then the unit of time is 2.17×10^{-12} s. The units of some of the physical quantities

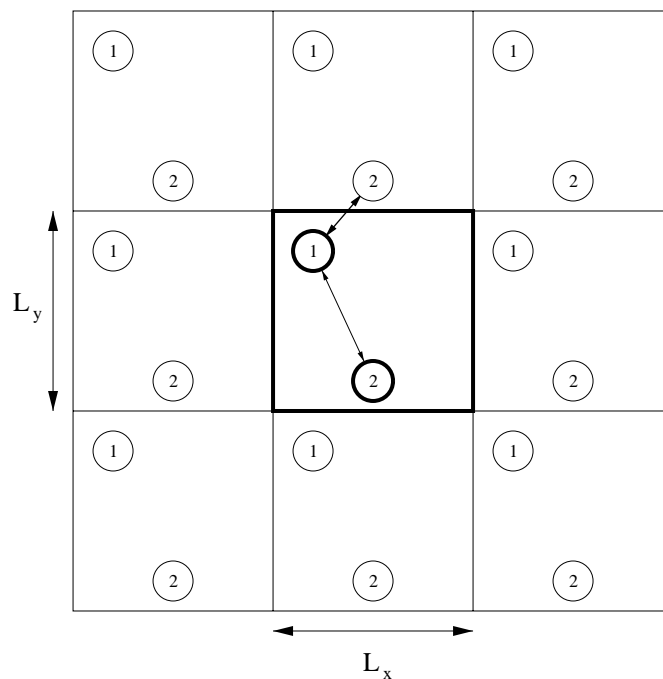


Figure 8.3: Example of the minimum image approximation in two dimensions. The minimum image distance convention implies that the separation between particles 1 and 2 is given by the shorter of the two distances shown.

of interest are shown in Table 8.1. All program variables are in reduced units, e.g., the time in our molecular dynamics program is expressed in units of $\sigma(m/\epsilon)^{1/2}$. As an example, suppose that we run our molecular dynamics program for 2000 time steps with a time step $\tau = 0.01$. The total time of our run is $2000 \times 0.01 = 20$ in reduced units or 4.34×10^{-11} s for argon (see Table 8.1). The total time of a typical molecular dynamics simulation is in the range of $10 - 10^4$ in reduced units, corresponding to a duration of approximately $10^{-11} - 10^{-9}$ s.

8.6 A Molecular Dynamics Program

In the following, we develop a molecular dynamics simulation of 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. The structure of Program md is given in the following:

```
PROGRAM md
PUBLIC x(36),y(36),vx(36),vy(36),ax(36),ay(36)
PUBLIC N,Lx,Ly,dt,dt2
```

quantity	unit	value for argon
length	σ	3.4×10^{-10} m
energy	ϵ	1.65×10^{-21} J
mass	m	6.69×10^{-26} kg
time	$\sigma(m/\epsilon)^{1/2}$	2.17×10^{-12} s
velocity	$(\epsilon/m)^{1/2}$	1.57×10^2 m/s
force	ϵ/σ	4.85×10^{-12} N
pressure	ϵ/σ^2	1.43×10^{-2} N · m ⁻¹
temperature	ϵ/k	120 K

Table 8.1: The system of units used in the molecular dynamics simulations of particles interacting via the Lennard-Jones potential. The numerical values of σ , ϵ , and m are for argon. The quantity k is Boltzmann's constant and has the value $k = 1.38 \times 10^{-23}$ J/K. The unit of pressure is for a two-dimensional system.

```

LIBRARY "csgraphics"
CALL initial(t,ke,kecum,pecum,vcum,area)
CALL set_up_windows(#1,#2)
CALL accel(pe,virial)
LET E = ke + pe           ! total energy
LET ncum = 0             ! number of times data accumulated
LET flag$ = ""
DO
  CALL show_positions(flag$,#2)
  CALL Verlet(t,ke,pe,virial)
  CALL show_output(t,ke,pe,virial,kecum,vcum,ncum,area,#1)
LOOP until flag$ = "stop"
CALL save_config
END

```

The x - and y -components of the positions, velocities, and accelerations are represented by arrays and are declared as public variables (cf. [Appendix 3C](#)) because they are used in almost all of the subroutines. These arrays are dimensioned in a `PUBLIC` statement and are declared in a `DECLARE PUBLIC` statement in each subroutine in which they are used. An array that is declared in a `PUBLIC` statement is not dimensioned in a `DIM` statement. The nature of the passed variables and the subroutines are discussed in the following.

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 do we choose the initial configuration (a set of positions and velocities) to correspond to a fluid at a desired temperature? We postpone a discussion of such questions until [Section 8.7](#), and instead we use initial conditions that have been computed previously.

The initial conditions can be incorporated into the program by either reading a data file or storing the information within a program using `DATA` and `READ` statements. The following program illustrates the use of the `DATA` and `READ` statements:

```

PROGRAM example_data
DIM x(6)
DATA 4.48,3.06,0.20,2.08,3.88,3.36
FOR i = 1 to 6
  READ x(i)           ! reads input from DATA statement
NEXT i
END

```

The location of the DATA statements is unimportant, but the data must be stored in the order in which it will be read by the READ statements.

The following version of SUB `initial` uses the DATA and READ statements to store an initial configuration of $N = 16$ particles in a central cell of linear dimension $L_x = L_y = 6$.

```

SUB initial(t,ke,kecum,pecum,vcum,area)
DECLARE PUBLIC x(),y(),vx(),vy()
DECLARE PUBLIC N,Lx,Ly,dt,dt2
DECLARE DEF pbc
LET dt = 0.01
LET dt2 = dt*dt
LET response$ = ""
DO
  INPUT prompt "read data statements (d) or file (f)? ": start$
  IF (start$ = "d") or (start$ = "D") then
    LET response$ = "ok"
    LET N = 16
    LET Lx = 6
    LET Ly = 6
    DATA 1.09,0.98,-0.33,0.78,3.12,5.25,0.12,-1.19
    DATA 0.08,2.38,-0.08,-0.10,0.54,4.08,-1.94,-0.56
    DATA 2.52,4.39,0.75,0.34,3.03,2.94,1.70,-1.08
    DATA 4.25,3.01,0.84,0.47,0.89,3.11,-1.04,0.06
    DATA 2.76,0.31,1.64,1.36,3.14,1.91,0.38,-1.24
    DATA 0.23,5.71,-1.58,0.55,1.91,2.46,-1.55,-0.16
    DATA 4.77,0.96,-0.23,-0.83,5.10,4.63,-0.31,0.65
    DATA 4.97,5.88,1.18,1.48,3.90,0.20,0.46,-0.51
    FOR i = 1 to N
      READ x(i),y(i),vx(i),vy(i)
    NEXT i
  ELSE IF (start$ = "f") or (start$ = "F") then
    LET response$ = "ok"
    INPUT prompt "file name = ": file$
    OPEN #1: name file$, access input
    INPUT #1: N
    INPUT #1: Lx
    INPUT #1: Ly
    INPUT #1: heading$

```

```

        FOR i = 1 to N
            INPUT #1: x(i),y(i)
        NEXT i
        INPUT #1: heading$
        FOR i = 1 to N
            INPUT #1: vx(i),vy(i)
        NEXT i
        CLOSE #1
    ELSE
        PRINT
        PRINT "d or f are the only acceptable responses."
    END IF
LOOP until response$ = "ok"
CLEAR
LET ke = 0                ! kinetic energy
FOR i = 1 to N
    LET ke = ke + vx(i)*vx(i) + vy(i)*vy(i)
NEXT i
LET ke = 0.5*ke
LET area = Lx*Ly
LET t = 0                ! time
! initialize sums
LET kecum = 0
LET pecum = 0
LET vcum = 0
END SUB

```

It is good programming practice to have an appropriate response for any input. For this reason SUB `initial` checks that an appropriate response is given to the INPUT statement.

SUB `accel` finds the total force on each particle and uses Newton's third law to reduce the number of calculations by a factor of two. (In reduced units, the mass of a particle is unity and hence acceleration and force are equivalent.) FUNCTION `separation` ensures that the separation between particles is no greater than $Lx/2$ in the x direction and $Ly/2$ in the y direction. The force on a given particle is assumed to be due to all the other $N - 1$ particles. Because of the short range nature of the Lennard-Jones potential, we could truncate the force at a distance $r = r_c \approx 2.5\sigma$ and ignore the forces from particles whose distance is greater than r_c . The meaning of the quantity `virial` in SUB `accel` is discussed in Section 8.7.

```

SUB accel(pe, virial)
    DECLARE PUBLIC x(), y(), ax(), ay()
    DECLARE PUBLIC N, Lx, Ly
    DECLARE DEF separation
    FOR i = 1 to N
        LET ax(i) = 0
        LET ay(i) = 0
    NEXT i

```

```

LET pe = 0
LET virial = 0
FOR i = 1 to N - 1          ! compute total force on particle i
  FOR j = i + 1 to N      ! due to particles j > i
    LET dx = separation(x(i) - x(j),Lx)
    LET dy = separation(y(i) - y(j),Ly)
    ! acceleration = force because mass = 1 in reduced units
    CALL force(dx,dy,fxij,fyij,pot)
    LET ax(i) = ax(i) + fxij
    LET ay(i) = ay(i) + fyij
    LET ax(j) = ax(j) - fxij  ! Newton's third law
    LET ay(j) = ay(j) - fyij
    LET pe = pe + pot
    LET virial = virial + dx*fx + dy*fy
  NEXT j
NEXT i
END SUB

FUNCTION separation(ds,L)
  IF ds > 0.5*L then
    LET separation = ds - L
  ELSE IF ds < -0.5*L then
    LET separation = ds + L
  ELSE
    LET separation = ds
  END IF
END DEF

SUB force(dx,dy,fx,fy,pot)
  LET r2 = dx*dx + dy*dy
  LET rm2 = 1/r2
  LET rm6 = rm2*rm2*rm2
  LET f_over_r = 24*rm6*(2*rm6 - 1)*rm2
  LET fx = f_over_r*dx
  LET fy = f_over_r*dy
  LET pot = 4*(rm6*rm6 - rm6)
END SUB

```

The Verlet algorithm for the numerical solution of Newton's equations of motion is implemented in SUB `Verlet`. Note that the velocity is partially updated using the old acceleration. Then SUB `accel` is called to determine the acceleration using the new position and the velocity is updated again. SUB `Verlet` calls FUNCTION `pbcc` which implements the periodic boundary conditions using IF statements.

```
SUB Verlet(t,ke,pe,virial)
```

```

DECLARE PUBLIC x(),y(),vx(),vy(),ax(),ay()
DECLARE PUBLIC N,Lx,Ly,dt,dt2
DECLARE DEF pbc
FOR i = 1 to N
  LET xnew = x(i) + vx(i)*dt + 0.5*ax(i)*dt2
  LET ynew = y(i) + vy(i)*dt + 0.5*ay(i)*dt2
  ! partially update velocity using old acceleration
  LET vx(i) = vx(i) + 0.5*ax(i)*dt
  LET vy(i) = vy(i) + 0.5*ay(i)*dt
  LET x(i) = pbc(xnew,Lx)
  LET y(i) = pbc(ynew,Ly)
NEXT i
CALL accel(pe,virial)          ! new acceleration
LET ke = 0
FOR i = 1 to N
  ! complete the update of the velocity using new acceleration
  LET vx(i) = vx(i) + 0.5*ax(i)*dt
  LET vy(i) = vy(i) + 0.5*ay(i)*dt
  LET ke = ke + vx(i)*vx(i) + vy(i)*vy(i)
NEXT i
LET ke = 0.5*ke
LET t = t + dt
END SUB

FUNCTION pbc(pos,L)
  IF pos < 0 then
    LET pbc = pos + L
  ELSE IF pos > L then
    LET pbc = pos - L
  ELSE
    LET pbc = pos
  END IF
END DEF
END DEF

```

In SUB `set_up_windows` we divide the screen into two windows so that the trajectories are shown in a separate window. One advantage of the use of multiple windows is that the `CLEAR` statement erases the current window only, rather than the entire screen.

```

SUB set_up_windows(#1,#2)
  DECLARE PUBLIC Lx,Ly
  OPEN #1: screen 0,1,0.90,1.0 ! numerical output
  OPEN #2: screen 0.02,1,0.02,0.90 ! particle trajectories
  CALL compute_aspect_ratio(Lx,xwin,ywin)
  SET WINDOW 0,xwin,0,ywin
  BOX LINES 0,Lx,0,Ly
  CALL headings(#1)

```

```

END SUB

SUB headings(#1)
  WINDOW #1
  SET CURSOR 1,1
  PRINT using "#####": "time steps";
  PRINT,
  PRINT using "###.##": "time";
  PRINT,
  PRINT using "---#.###": "energy";
  PRINT,
  PRINT using "###.##": "<T>";
  PRINT,
  PRINT using "###.##": "<P>"
END SUB

```

The kinetic energy is computed at each time step and its values are accumulated in `SUB show_output`. Although it is inefficient to compute the kinetic energy (and the virial) at every time step, we do so for simplicity.

```

SUB show_output(t,ke,pe,virial,kecum,vcum,ncum,area,#1)
  WINDOW #1
  DECLARE PUBLIC N,Lx,Ly
  SET CURSOR 2,1
  SET COLOR "black/white"
  LET ncum = ncum + 1
  PRINT using "#####": ncum;
  PRINT,
  PRINT using "###.##": t;      ! time
  PRINT,
  LET E = ke + pe              ! total energy
  PRINT using "---#.###": E;
  PRINT,
  LET kecum = kecum + ke
  LET vcum = vcum + virial
  LET mean_ke = kecum/ncum    ! still need to divide by N
  LET p = mean_ke + (0.5*vcum)/ncum ! mean pressure * area
  LET p = p/area
  PRINT using "###.##": mean_ke/N; ! mean kinetic temperature
  PRINT,
  PRINT using "###.##": p;
END SUB

```

The trajectories of the individual particles are displayed in `SUB show_positions` by representing the position of a particle at each time step by a point. The trajectories eventually fill the screen, which can be cleared by pressing the letter 'r' (refresh). If the letter 'n' is pressed, the

particle positions are not shown on the screen and the program runs faster. Pressing 'r' restarts the screen updates. The program can be stopped by pressing the letter 's' (stop).

```

SUB show_positions(flag$,#2)
  DECLARE PUBLIC x(),y(),N,Lx,Ly
  IF key input then
    GET KEY k
    IF k = ord("r") then
      WINDOW #2
      CLEAR
      SET COLOR "black"
      BOX LINES 0,Lx,0,Ly
      LET flag$ = ""
    ELSEIF k = ord("s") then
      LET flag$ = "stop"
    ELSEIF k = ord("n") then
      LET flag$ = "no_show"
    END IF
  END IF
  IF flag$ <> "no_show" then
    WINDOW #2
    SET COLOR "red"
    FOR i = 1 to N
      PLOT x(i),y(i)
    NEXT i
  END IF
END SUB

```

Frequently, it is convenient to start a new run from the last configuration of a previous run. The final configuration is saved to a file in the following:

```

SUB save_config
  DECLARE PUBLIC x(),y(),vx(),vy()
  DECLARE PUBLIC N,Lx,Ly
  INPUT prompt "file name of saved configuration = ": file$
  OPEN #1: name file$,access output,create new
  PRINT #1: N
  PRINT #1: Lx
  PRINT #1: Ly
  PRINT #1: "x(i)","y(i)"
  ! comma added between outputs on the same line so that file
  ! can be read by True BASIC
  FOR i = 1 to N
    PRINT #1, using "----.####, ----.####": x(i),y(i)
  NEXT i
  PRINT #1: "vx(i)","vy(i)"

```

```

FOR i = 1 to N
  PRINT #1, using "----.###, ----.####": vx(i),vy(i)
NEXT i
CLOSE #1
END SUB

```

In Problem 8.2 we use this preliminary version of Program md to simulate the approach of a system to equilibrium.

Problem 8.2. Approach to equilibrium

1. The initial configuration incorporated into the DATA statements in Program md corresponds to $N = 16$ particles interacting via the Lennard-Jones potential in a square cell of linear dimension $L = 6$. Check that the x and y coordinates of all the particles lies between 0 and 6. Set $\Delta t = 0.01$ and run the program to make sure that it is working properly. The total energy should be approximately conserved and the trajectories of all sixteen particles should be seen on the screen.
2. Suppose that at $t = 0$, the constraint that $0 \leq x \leq 6$ is removed and the particles can move freely in a rectangular cell with $L_x = 12$ and $L_y = 6$. Incorporate this change into your program and observe the trajectories of the particles. Does the system become more or less random as time increases?
3. Compute $n(t)$, the number of particles in the left half of the cell, and plot $n(t)$ as a function of t . What is the qualitative behavior of $n(t)$? Also compute the time average of $n(t)$, and plot it as a function of t . What is the mean number of particles on the left half after the system has reached equilibrium? Compare your qualitative results with the results you found in Problem 7.1. Would the approach to equilibrium be better defined if you did a molecular dynamics simulation with $N = 64$ particles?

Problem 8.3. Sensitivity to initial conditions

1. Consider the following initial condition corresponding to $N = 11$ particles moving in the same direction with the same velocity (see Figure 8.4). Choose $L_x = L_y = 10$ and $\Delta t = 0.01$.

```

FOR i = 1 to N
  LET x(i) = Lx/2
  LET y(i) = (i - 0.5)*Ly/N
  LET vx(i) = 1
  LET vy(i) = 0
NEXT i

```

Does the system eventually reach equilibrium? Why or why not?

2. Change the velocity of particle 6 so that $v_x(6) = 0.99$ and $v_y(6) = 0.01$. Is the behavior of the system qualitatively different than in part (a)? Does the system eventually reach equilibrium? Are the trajectories of the particles sensitive to the initial conditions? Explain why almost all initial states lead to the same qualitative behavior.

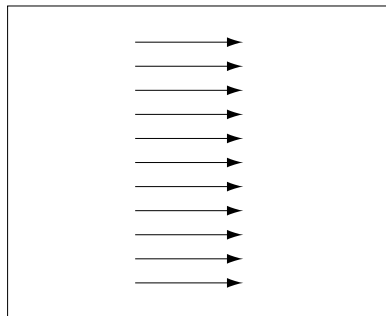


Figure 8.4: Example of a special initial condition; the arrows represent the magnitude and the direction of each particle's velocity.

3. Modify `Program md` so that the program runs for a predetermined time interval. Use the initial condition included in the `DATA` statements, and save the positions and velocities of all the particles at $t = 0.5$ in a file. Then consider the time reversed process, i.e., the motion that would occur if the direction of time were reversed. This reversal is equivalent to letting $\mathbf{v} \rightarrow -\mathbf{v}$ for all particles. Do the particles return to their original positions? What happens if you reverse the velocities at a later time? What happens if you choose a smaller value of Δt ?
4. Use the initial condition included in the `DATA` statements, but let $L_x = 12$. Save the positions and velocities of all the particles in a file at $t = 2.5$. Then consider the time-reversed process. Do the particles return to their original state? What happens if you reverse the velocities at $t = 5$ rather than at $t = 2.5$?
5. What can you conclude about the chaotic nature of the trajectories from your results? Are the computed trajectories the same as the “true” trajectories?

The trajectories generated in Problems 8.2 and 8.3 show the system in its greatest possible detail. From this *microscopic* viewpoint, the trajectories appear rather complex. The system can be described more simply by specifying its *macroscopic state*. For example, in Problem 8.2 we described the approach of the system to equilibrium by specifying n , the number of particles in the left half of the cell. Your observations of the macroscopic variable $n(t)$ should be consistent with the general properties of many body systems:

1. After the removal of an internal constraint, an isolated system changes in time from a “less random” to a “more random” state.
2. The equilibrium macroscopic state is characterized by relatively small fluctuations about a mean that is independent of time. A many particle system whose macroscopic state is independent of time is said to be in *equilibrium*. (The relative fluctuations become smaller as the number of particles becomes larger.)

In Problem 8.2(b) and (c) we found that the particles filled the box, and hence we were able to define a direction of time. Of course, this direction would be better defined if we considered

more particles. Note that there is nothing intrinsic in Newton's laws of motion that gives time a preferred direction.

Before we consider other macroscopic variables, we need to monitor the total energy and verify our claim that the Verlet algorithm maintains conservation of energy with a reasonable choice of Δt . We also introduce a check for momentum conservation.

Problem 8.4. Tests of the Verlet algorithm

1. One essential check of a molecular dynamics program is that the total energy be conserved to the desired accuracy. Use `Program md` and determine the value of Δt necessary for the total energy to be conserved to a given accuracy over a time interval of $t = 2$. One criterion is to compute $\Delta E_{\max}(t)$, the maximum of the difference $|E(t) - E(0)|$, over the time interval t , where $E(0)$ is the initial total energy, and $E(t)$ is the total energy at time t . Verify that $\Delta E_{\max}(t)$ decreases when Δt is made smaller for fixed t . If your program is working properly, then $\Delta E_{\max}(t)$ should decrease as approximately $(\Delta t)^2$.
2. Because of the use of periodic boundary conditions, all points in the central cell are equivalent and the system is translationally invariant. Explain why `Program md` should conserve the total linear momentum. Floating point error and the truncation error associated with a finite difference algorithm can cause the total linear momentum to drift. Programming errors also might be detected by checking for the conservation of momentum. Hence, it is a good idea to monitor the total linear momentum at regular intervals and reset the total momentum equal to zero if necessary. `SUB check_momentum`, listed in the following, should be called in `SUB initial` after the initial configuration is obtained and in the main loop of the program at regular intervals, e.g., every 100 – 1000 time steps. How well does `Program md` conserve the total linear momentum for $\Delta t = 0.01$?

```

SUB check_momentum
  DECLARE PUBLIC vx(),vy(),N
  LET vxsum = 0
  LET vysum = 0
  ! compute total center of mass velocity (momentum)
  FOR i = 1 to N
    LET vxsum = vxsum + vx(i)
    LET vysum = vysum + vy(i)
  NEXT i
  LET vxcm = vxsum/N
  LET vycm = vysum/N
  FOR i = 1 to N
    LET vx(i) = vx(i) - vxcm
    LET vy(i) = vy(i) - vycm
  NEXT i
END SUB

```

3. Another way of monitoring how well the program is conserving the total energy is to analyze the time series $E(t)$ using a least squares fit of $E(t)$ to a straight line. The slope of the line can be interpreted as the drift and the root mean square deviation from the straight line

can be interpreted as the noise (σ_y in the notation of Section 7.5). How does the drift and the noise depend on Δt for a fixed time interval t ? Most research applications conserve the energy to within 1 part in 10^3 or 10^4 or better over the duration of the run.

4. Consider one of the higher-order algorithms discussed in Appendix 5A for the solution of Newton's equations of motion. Can you choose a larger value of Δt to achieve the same degree of energy conservation that you found using the Verlet algorithm? Does the total energy fluctuate and/or eventually drift?

8.7 Thermodynamic Quantities

In the following we discuss how some of the macroscopic quantities of interest such as the temperature and the pressure can be related to time averages over the phase space trajectories of the particles. We then use our molecular dynamics program to explore the qualitative properties of gases and liquids.

The kinetic definition of the temperature follows from the equipartition theorem: each quadratic term in the energy of a classical system in equilibrium at temperature T has a mean value equal to $\frac{1}{2}kT$. Hence, we can define the temperature $T(t)$ at time t by the relation

$$N \frac{d}{2} kT(t) = K(t), \quad (8.5)$$

or

$$kT(t) = \frac{2}{d} \frac{K(t)}{N} = \frac{1}{dN} \sum_{i=1}^N m_i \mathbf{v}_i(t) \cdot \mathbf{v}_i(t). \quad (8.6)$$

K represents the total kinetic energy of the system, d is the spatial dimension, and the sum is over the N particles in the system. The mean temperature can be expressed as the time average of $T(t)$ over many configurations of the particles. For two dimensions ($d = 2$) and our choice of units, we write the mean temperature T as

$$T \equiv \bar{T} = \frac{1}{2N} \sum_{i=1}^N \overline{m_i \mathbf{v}_i(t) \cdot \mathbf{v}_i(t)}. \quad (\text{two dimensions}) \quad (8.7)$$

The relation (8.7) is an example of the relation of a macroscopic quantity, the mean temperature, to a time average over the trajectories of the particles.

Note that the relation (8.6) holds only if the momentum of the center of mass of the system is zero — we do not want the motion of the center of mass to change the temperature. In a laboratory system the walls of the container ensure that the center of mass motion is zero (if the mean momentum of the walls is zero). In our simulation, we impose the constraint that the center of mass momentum (in d directions) be zero. Consequently, the system has $dN - d$ independent velocity components rather than dN components, and we should replace (8.6) by

$$kT(t) = \frac{1}{dN - d} \sum_{i=1}^N m_i \mathbf{v}_i(t) \cdot \mathbf{v}_i(t). \quad (8.8)$$

The presence of the factor $d(N - 1)$ rather than dN in (8.8) is an example of a **finite size** correction which becomes unimportant for large N . We shall ignore this correction in the following.

Another macroscopic quantity of interest is the mean pressure of the system. The pressure is related to the force per unit area acting normal to an imaginary surface in the system. By Newton's second law, this force can be related to the momentum that crosses the surface per unit time. In general, this momentum flux has two contributions. The easiest contribution to understand is the one carried by the particles due to their motion. This contribution, equal to the pressure of an ideal gas, is derived in many texts (cf. Chapter 7 of Reif) using simple kinetic theory arguments and is given by $P_{\text{ideal}} = NkT/V$.

The other contribution to the momentum flux arises from the momentum transferred across the surface due to the forces between particles on different sides of the surface. The form of this contribution to the dynamical pressure is difficult to derive if periodic boundary conditions are used (cf. Haile). The instantaneous pressure at time t including both contributions to the momentum flux is given by

$$P(t) = \frac{N}{V}kT(t) + \frac{1}{dV} \sum_{i < j} \mathbf{r}_{ij}(t) \cdot \mathbf{F}_{ij}(t), \quad (8.9)$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$, and \mathbf{F}_{ij} is the force on particle i due to particle j .

The mean pressure $P \equiv \overline{P(t)}$ is found by computing a time average of the right-hand side of (8.9). The computed quantity of interest is not P , but the quantity

$$\frac{PV}{NkT} - 1 = \frac{1}{dNkT} \sum_{i < j} \overline{\mathbf{r}_{ij} \cdot \mathbf{F}_{ij}}. \quad (8.10)$$

In **Program md**, the right-hand side of (8.10), known as the **virial**, is computed in **SUB accel** and accumulated in the variable **vcum**. This quantity represents the correction to the ideal gas equation of state due to interactions between the particles.

The relation of information at the microscopic level to macroscopic quantities such as the temperature and pressure is one of the fundamental elements of statistical mechanics. In brief, molecular dynamics allows us to compute various time averages of the phase space trajectory over finite time intervals. The main practical question we must consider is whether our time intervals are sufficiently long to allow the system to explore phase space and yield meaningful averages. In equilibrium statistical mechanics, a time average is replaced by an **ensemble** average over all possible configurations. The quasi-ergodic hypothesis asserts the equivalence of these two averages if the same quantities are held fixed. In statistical mechanics, the ensemble of systems at fixed E, V , and N is called the microcanonical ensemble. Averages in this ensemble correspond to the time averages we find in molecular dynamics which are at fixed E, V and N . (Molecular dynamics also imposes an additional, but unimportant, constraint on the center of mass motion.) Ensemble averages are explored using Monte Carlo methods in Chapters ?? and ??.

The goal of the following problems is to explore some of the qualitative features of gases, liquids, and solids. Because we consider only small systems and relatively short run times, our results will only be suggestive.

Problem 8.5. Qualitative properties of a liquid and a gas

1. Use the initial configuration stored in the DATA statements in Program md for this problem. (If you do not want to type in the initial conditions, see Problem 8.11 for a discussion of how to generate your own.) For this initial condition $N = 16$ and $L_x = L_y = 6$. What is the reduced density? What is the initial energy of the system? Choose $\Delta t = 0.01$ and run the simulation for at least 500 time steps or $t = 5$. Compare your estimate for P with the value for an ideal gas.
2. Modify your program so that the instantaneous values of the temperature and pressure are not accumulated until the system has reached equilibrium. What is your criterion for equilibrium? One criterion is to compute the average values of T and P over finite time intervals and check that these averages do not drift with time.
3. One way of starting a simulation is to use the positions saved from an earlier run. The simplest way of obtaining an initial condition corresponding to a different density, but with the same value of N , is to rescale the positions of the particles and the linear dimensions of the cell. The following code shows one way to do so.

```

LET rscale = 0.95
FOR i = 1 to N
    LET x(i) = rscale*x(i)
    LET y(i) = rscale*y(i)
NEXT i
LET Lx = rscale*Lx
LET Ly = rscale*Ly
LET area = Lx*Ly

```

Incorporate the above code into your program in a separate subroutine. How do you expect P and T to change when the system is compressed? Use the same initial configuration as in part (a) and determine the mean temperature and pressure for the density $\rho = 16/(5.7)^2 \approx 0.49$ (`rescale = 0.95`). Compare your result for P to the ideal gas result. What do you think would happen if you choose a value of `rescale` that is much smaller? Save the final configuration of your simulation in a file and use it as the initial condition for another run with $\rho = 16/(0.95 \times 5.7)^2 \approx 0.55$. Determine T and P for this higher density and save the final configuration.

Problem 8.6. Distribution of speeds and velocities

1. Write a subroutine to compute the equilibrium probability $P(v)\Delta v$ that a particle has a speed between v and $v + \Delta v$. To do so, estimate the value of the maximum speed `vmax` that you need to bin. Choose bins of width $dv = \text{vmax}/\text{nbins}$, where `nbins` is the total number of bins. The following code illustrates one way of putting the speeds into their proper bins:

```

LET v2 = vx(i)*vx(i) + vy(i)*vy(i)
LET v = sqr(v2)
LET ibin = truncate(v/dv,0) + 1
IF ibin > nbins then LET ibin = nbins
LET prob(ibin) = prob(ibin) + 1

```

The array element `prob(ibin)` records the number of times the speed of a particle corresponds to `ibin`. Although it is inefficient, determine `prob(ibin)` after every time step for at least 100 time steps. Normalize `prob(ibin)` by dividing by the number of particles and by the number of time steps. Use the initial configuration stored in the `DATA` statements in Program `md` for this problem. Choose `nbin = 50`.

2. Plot the probability density $P(v)$ versus v . What is the qualitative form of $P(v)$? What is the most probable value of v ? What is the approximate width of $P(v)$? Compare your measured result to the theoretical form (in two dimensions)

$$P(v) dv = A e^{-mv^2/2kT} v dv. \quad (8.11)$$

The form (8.11) of the distribution of speeds is known as the Maxwell-Boltzmann distribution.

3. Determine the probability density for the x and y components of the velocity. Make sure that you distinguish between positive and negative values. What is the most probable value for the x and y velocity components? What are their average values? Plot the probability densities $P(v_x)$ versus v_x and $P(v_y)$ versus v_y . Better results can be found by plotting the average $[P(v_x = w) + P(v_y = w)]/2$ versus w . What is the qualitative form of $P(\mathbf{v})$? Is it the same as the probability density for the speed?

Another useful thermal quantity is the **heat capacity** at constant volume defined by the relation $C_V = (\partial E / \partial T)_V$. C_V is an example of a linear response function, that is, the response of the temperature to energy in the form of heat added to the system.

One way to obtain C_V is to determine $T(E)$, the temperature as a function of E . (Remember that T is obtained as a function of E in the microcanonical ensemble.) The heat capacity is given by $\Delta E / \Delta T$ for two runs that have slightly different temperatures. This method is straightforward, but requires that simulations at different energies be done before the derivative can be estimated. An alternative way of estimating C_V from the fluctuations of the kinetic energy is discussed in Problem 8.7.

Problem 8.7. Energy dependence of the temperature and pressure

1. We have seen in Problem 8.5 that the total energy is determined by the initial conditions, and the temperature is a derived quantity found only after the system has reached thermal equilibrium. For this reason it is difficult to study the system at a particular temperature. The mean temperature can be changed to the desired temperature by rescaling the velocities of the system, but we have to be careful not to increase the velocities too quickly. Why? Choose the initial configuration of the system to be an equilibrium configuration from an earlier simulation for $L_x = L_y = 6$ and $N = 16$ and determine $T(E)$, the energy dependence of the mean temperature, in the range $T = 1.0$ to $T = 1.2$. Rescale the velocities by the desired factor. Is the equilibrium temperature increased to the desired value? If not, you need to rescale the velocities again. In general, the desired temperature is reached by a series of velocity rescalings over a sufficiently long time such that the system remains close to equilibrium during the rescaling.
2. Use your data for $T(E)$ found in part (a) to plot the total energy E as a function of T . Is T a monotonically increasing function of E ? Estimate the contribution to C_V from the potential

energy. What percentage of the contribution to the heat capacity is due to the potential energy? Why are accurate determinations of C_V difficult to achieve?

3. In our molecular dynamics simulations, the total energy is fixed, but the kinetic and potential energies can fluctuate. Another way of determining C_V is to relate it to the fluctuations of the kinetic energy. (In Chapter ?? we find that C_V is related to the fluctuations of the total energy in the constant T, V, N ensemble.) It can be shown that (cf. Ray and Graben)

$$\overline{T^2} - \bar{T}^2 = \frac{d}{2} N (k\bar{T})^2 \left[1 - \frac{dNk}{2C_V} \right]. \quad (8.12)$$

Note that the relation (8.12) reduces to the ideal gas result if $\overline{T^2} = \bar{T}^2$. Write a subroutine to determine C_V from (8.12) and compare your results with the determination of C_V in part (b). What are the advantages and disadvantages of determining C_V from the fluctuations of T compared to the method used in part (b)?

How can we generate a typical initial condition for a system of particles at high density? What happens if we simply place the particles at random in the central cell? The problem is that if the system is dense, some particles will be very close to one another and exert a very large repulsive force F on one another. Because the condition $(F/m)(\Delta t)^2 \ll \sigma$ must be satisfied for a finite difference method to be applicable, the random placement of particles is not practical. However, it is possible to use such an initial condition if a fictitious drag force proportional to the square of the velocity is introduced to equilibrate the system. The effect of such a force is to damp the velocity of those particles whose velocities become too large due to the large forces exerted on them.

In general, the best way of obtaining a configuration with the desired density and number of particles is to place the particles on the sites of a regular lattice. If the goal is to equilibrate the system at fluid densities, the symmetry of the lattice is not important. The initial velocities can be chosen at random according to the Maxwell-Boltzmann distribution you found in Problem 8.6c. However, as you have found, the velocities equilibrate very quickly, and we may simply choose each component of \mathbf{v}_i with uniform probability such that the desired temperature is obtained. Because the velocities are chosen at random, we need to use `SUB check_momentum` to ensure that the initial total momentum in the x and y direction is zero. After the system has come into partial equilibrium, you can increase the velocities of all the particles to reduce the time it takes to “melt” the system and reach the desired fluid state.

To simulate a solid we need to choose the shape of the central cell to be consistent with the symmetry of the solid phase of the system. This choice is necessary even though we have used periodic boundary conditions to minimize surface effects. If the cell does not correspond to the correct crystal structure, the particles cannot form a perfect crystal, and some of the particles will wander around in an endless search for their “correct” position. Consequently, a simulation of a small number of particles at high density and low temperature would lead to spurious results.

We know that the equilibrium structure of a crystalline solid at $T = 0$ is the configuration of lowest energy. In Problem 8.8 we compute the energy of a Lennard-Jones solid for both the square and triangular lattice structures (see Figure 8.5).

Problem 8.8. Ground state energy of two-dimensional lattices The nature of the triangular lattice can be seen from Figure 8.5. Each particle has six nearest neighbors. Although it is possible to

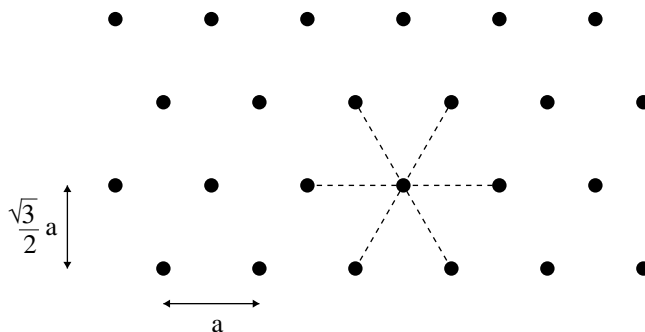


Figure 8.5: Each particle has six nearest neighbors in a triangular lattice.

choose the central cell of the triangular lattice to be a rhombus, it is more convenient to choose the cell to be rectangular. We take the linear dimensions of the cell to be L_x and $L_y = \sqrt{3}L_x/2$ respectively. For simplicity, we assume that \sqrt{N} is an integer so that the lattice spacings in the horizontal and vertical directions are $a_x = L_x/\sqrt{N}$ and $a_y = L_y/\sqrt{N}$, respectively. The lattice sites in each row are displaced by $\frac{1}{2}a_x$ from the preceding row. The following code generates a triangular lattice.

```

LET i = 0
FOR col = 1 to nx
  FOR row = 1 to ny
    LET i = i + 1
    IF (mod(row,2) = 0) then
      LET x(i) = (col - 0.75)*ax
    ELSE
      LET x(i) = (col - 0.25)*ax
    END IF
    LET y(i) = (row - 0.5)*ay
  NEXT row
NEXT col

```

Write a program to compute the potential energy per particle of a system of N particles interacting via the Lennard-Jones potential. Consider both the triangular and square lattices, and choose the linear dimension of the square lattice to be $L = \sqrt{L_x L_y}$, so that both lattices have the same density. Choose $N = 36$ and determine the energy for $L_x = 5$ and $L_x = 7$. What is the density of the system for each case? Do your results for E/N depend on the size of the lattice? Which lattice symmetry has a lower energy? Explain your results in terms of the ability of the triangular lattice to pack the particles closer together.

Problem 8.9. The solid state and melting

1. Choose $N = 16$, $L_x = 4$, and $L_y = \sqrt{3}L_x/2$, and place the particles on a triangular lattice. Give each particle zero initial velocity. What is the total energy of the system? Do a

simulation and measure the temperature and pressure as a function of time. Does the system remain a solid?

2. Give the particles a random velocity in the interval $[-0.5, +0.5]$. What is the total energy? Equilibrate the system and determine the mean temperature and pressure. Describe the trajectories of the particles. Are the particles localized? Is the system a solid? Save an equilibrium configuration for use in part (c).
3. Choose the initial configuration to be an equilibrium configuration from part (b), but increase the kinetic energy by a factor of two. What is the new total energy? Describe the qualitative behavior of the motion of the particles. What is the equilibrium temperature and pressure of the system? After equilibrium is reached, increase the temperature again by rescaling the velocities in the same way. Repeat this rescaling and measure $P(T)$ and $E(T)$ for several different temperatures.
4. Use your results from part (c) to plot $E(T) - E(0)$ and $P(T)$ as a function of T . Is the difference $E(T) - E(0)$ proportional to T ? What is the mean potential energy for a harmonic solid? What is its heat capacity?
5. Choose an equilibrium configuration from part (b) and decrease the density by rescaling L_x , L_y and the particle coordinates by a factor of 1.1. What is the nature of the trajectories? Decrease the density of the system until the system melts. What is your qualitative criterion for melting?

It is possible for a system to not be in equilibrium even though its properties do not change over a long time interval. For example, if we rapidly lower the temperature of a liquid below its freezing temperature, it is likely that the resulting state is not an equilibrium crystal, but rather a supercooled liquid that will eventually nucleate to a crystal. In general, we must carefully prepare our system so as to minimize the probability that the system becomes trapped in a **metastable state**. On the other hand, there is much interesting physics and many applications to material science in the kinetics of nonequilibrium systems as they evolve toward equilibrium.

Problem 8.10. Metastability

1. We can create a metastable state by placing the particles in a square cell whose shape is not consistent with the lowest energy state corresponding to a triangular lattice. Modify `SUB initial` so that the particle positions form a square lattice. If the initial velocities are set to zero, what happens when you run the program? Choose $N = 64$ and $L_x = L_y = 9$.
2. We can show that the system in part (a) is in a metastable state by giving the particles a small random initial velocity with `vmax = 0.5`. Does the square lattice immediately change? When do you begin to see local structure resembling a triangular lattice?
3. If time permits, repeat part (b) with `vmax = 0.1`.

8.8 Radial Distribution Function

We can gain more insight into the structure of a many body system by looking at how the positions of the particles are correlated with one another due to their interactions. The **radial**

distribution function $g(r)$ is the most common measure of this correlation and is defined as follows. Suppose that N particles are contained in a region of volume V with number density $\rho = N/V$. (In two and one dimensions, we replace V by the area and length respectively.) Choose one of the particles to be the origin. Then the mean number of other particles in the shell between \mathbf{r} and $\mathbf{r} + d\mathbf{r}$ is given by $\rho g(\mathbf{r}) d\mathbf{r}$, where the volume element $d\mathbf{r} = 4\pi r^2 dr$ ($d = 3$), $2\pi r dr$ ($d = 2$), or $2 dr$ ($d = 1$). If the interparticle interaction is spherically symmetric and the system is a gas or a liquid, then $g(r)$ depends only on the separation $r = |\mathbf{r}|$. The normalization condition for $g(r)$ is

$$\rho \int g(r) d\mathbf{r} = N - 1 \approx N. \quad (8.13)$$

Equation (8.13) implies that if we choose one particle as the origin and count all the other particles in the system, we obtain $N - 1$ particles. For an ideal gas, there are no correlations between the particles, and $g(r) = 1$ for all r . For the Lennard-Jones interaction, we expect that $g(r) \rightarrow 0$ as $r \rightarrow 0$, because the particles cannot penetrate one another. We also expect that $g(r) \rightarrow 1$ as $r \rightarrow \infty$, because the effect of one particle on another decreases as their separation increases.

The radial distribution function can be measured indirectly by elastic radiation scattering experiments, especially by the scattering of X-rays. Several thermodynamic properties also can be obtained from $g(r)$. Because $\rho g(r)$ can be interpreted as the local density about a given particle, the potential energy of interaction between this particle and all other particles between r and $r + dr$ is $u(r)\rho g(r) d\mathbf{r}$, if we assume that only two-body interactions are present. The total potential energy is found by integrating over all values of r and multiplying by $N/2$. The factor of N is included because any of the N particles could be chosen as the particle at the origin, and the factor of $1/2$ is included so that each pair interaction is counted only once. The result is that the mean potential energy per particle can be expressed as

$$\frac{U}{N} = \frac{\rho}{2} \int g(r) u(r) d\mathbf{r}. \quad (8.14)$$

It also can be shown that the relation (8.10) for the mean pressure can be rewritten in terms of $g(r)$ so that the equation of state can be expressed as

$$\frac{PV}{NkT} = 1 - \frac{\rho}{2dkT} \int g(r) r \frac{du(r)}{dr} d\mathbf{r}. \quad (8.15)$$

To determine $g(r)$ for a particular configuration of particles, we first compute $n(r, \Delta r)$, the number of particles in a spherical (circular) shell of radius r and small, but nonzero width Δr , with the center of the shell centered about each particle. A subroutine for computing $n(r)$ is given in the following:

```

SUB compute_g(ncorrel)
  DECLARE PUBLIC x(),y()
  DECLARE PUBLIC N,Lx,Ly
  DECLARE PUBLIC gcum(),nbin,dr
  DECLARE DEF separation
  ! accumulate data for n(r)
  FOR i = 1 to N - 1

```

```

FOR j = i + 1 to N
  LET dx = separation(x(i) - x(j),Lx)
  LET dy = separation(y(i) - y(j),Ly)
  LET r2 = dx*dx + dy*dy
  LET r = sqr(r2)
  LET ibin = truncate(r/dr,0) + 1
  IF ibin <= nbin then
    LET gcum(ibin) = gcum(ibin) + 1
  END IF
NEXT j
NEXT i
LET ncorrel = ncorrel + 1      ! # times n(r) computed
END SUB

```

The results for $n(r)$ for different configurations are accumulated in the array `gcum`; the latter array is normalized in `SUB normalize_g` listed below. The use of periodic boundary conditions in `SUB compute_g` implies that the maximum separation between any two particles in the x and y direction is $Lx/2$ and $Ly/2$ respectively. Hence for a square cell, we can determine $g(r)$ only for $r \leq \frac{1}{2}L$.

To obtain $g(r)$ from $n(r)$, we note that for a given particle i , we consider only those particles whose j is greater than i (see `SUB compute_g`). Hence, there are a total of $\frac{1}{2}N(N-1)$ separations that are considered. In two dimensions we compute $n(r, \Delta r)$ for a circular shell whose area is $2\pi r \Delta r$. These considerations imply that $g(r)$ is related to $n(r)$ by

$$\rho g(r) = \frac{\overline{n(r, \Delta r)}}{\frac{1}{2}N 2\pi r \Delta r}. \quad (\text{two dimensions}) \quad (8.16)$$

Note the factor of $N/2$ in the denominator of (8.16). The following subroutine normalizes the array `gcum` and yields $g(r)$:

```

SUB normalize_g(ncorrel)
  DECLARE PUBLIC N,Lx,Ly
  DECLARE PUBLIC gcum(),dr
  LET density = N/(Lx*Ly)
  LET rmax = min(Lx/2,Ly/2)
  LET normalization = density*ncorrel*0.5*N
  LET bin = 1
  LET r = 0
  OPEN #2: name "gdata", access output,create new
  DO while r <= rmax
    LET area_shell = pi*((r + dr)^2 - r^2)
    LET g = gcum(bin)/(normalization*area_shell)
    PRINT r+dr/2,g
    PRINT #2: r+dr/2,g
    LET bin = bin + 1
    LET r = r + dr
  LOOP

```

CLOSE #2
END SUB

The shell thickness Δr needs to be sufficiently small so that the important features of $g(r)$ are found, but large enough so that each bin has a reasonable number of contributions. The value of Δr can be specified in SUB `initial`; a reasonable compromise choice for its magnitude is $\text{dr} = 0.025$.

Problem 8.11. The structure of $g(r)$ for a dense liquid and a solid

1. Incorporate SUB `compute_g` and SUB `normalize_g` into your molecular dynamics program and determine $g(r)$ for some of the same densities and temperatures that you have considered in previous problems. What are the qualitative features of $g(r)$?
2. Compute $g(r)$ for a system of $N = 64$ particles that are fixed on a triangular lattice with $L_x = 8$ and $L_y = \sqrt{3}L_x/2$. What is the density of the system? What is the nearest neighbor distance between sites? At what value of r does the first maximum of $g(r)$ occur? What is the next nearest distance between sites? At what value of r does the second maximum of $g(r)$ occur? Does your calculated $g(r)$ have any other relative maxima? If so, relate these maxima to the structure of the triangular lattice.
3. Use your molecular dynamics program to compute $g(r)$ for a dense fluid ($\rho > 0.6$, $T \approx 1.0$) using at least $N = 32$ particles. How many relative maxima can you observe? In what ways do they change as the density is increased? How does the behavior of $g(r)$ for a dense liquid compare to that of a dilute gas and a solid?

8.9 Hard disks

How can we understand the temperature and density dependence of the equation of state and the structure of a dense liquid? One way to gain more insight is to modify the interaction and see how the properties of the system change. In particular, we would like to understand the relative role of the repulsive and attractive parts of the interaction. For this reason, we consider an idealized system of hard disks for which the interaction $u(r)$ is purely repulsive:

$$u(r) = \begin{cases} +\infty, & r < \sigma \\ 0, & r \geq \sigma. \end{cases} \quad (8.17)$$

The length σ is the diameter of the hard disks (see Figure 8.6). In three dimensions the interaction (8.17) describes the interaction of hard spheres (billiard balls); in one dimension (8.17) describes the interaction of hard rods.

Because the interaction $u(r)$ between hard disks is a discontinuous function of r , the dynamics of hard disks is qualitatively different than it is for a continuous interaction such as the Lennard-Jones potential. For hard disks, the particles move in straight lines at constant speed between collisions and change their velocities instantaneously when a collision occurs. Hence the problem becomes finding the next collision and computing the change in the velocities of the colliding pair. We will see that the dynamics can be computed exactly in principle and is limited only by computer roundoff errors.

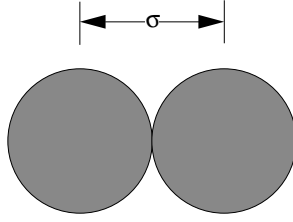


Figure 8.6: The closest distance between two hard disks is σ . The disks exert no force on one another unless they touch.

The dynamics of a system of hard disks can be treated as a sequence of two-body elastic collisions. The idea is to consider all pairs of particles i and j and to find the collision time t_{ij} for their next collision ignoring the presence of all other particles. In many cases, the particles will be going away from each other and the collision time is infinite. From the collection of collision times for all pairs of particles, we find the minimum collision time. We then move all particles forward in time until the collision occurs and calculate the postcollision velocities of the colliding pair.

We first determine the particle velocities after a collision. Consider a collision between particles 1 and 2. Let \mathbf{v}_1 and \mathbf{v}_2 be their velocities before the collision and \mathbf{v}'_1 and \mathbf{v}'_2 be their velocities after the collision. Because the particles have equal mass, it follows from conservation of energy and linear momentum that

$$v_1'^2 + v_2'^2 = v_1^2 + v_2^2 \quad (8.18)$$

$$\mathbf{v}'_1 + \mathbf{v}'_2 = \mathbf{v}_1 + \mathbf{v}_2. \quad (8.19)$$

From (8.19) we have

$$\Delta\mathbf{v}_1 = \mathbf{v}'_1 - \mathbf{v}_1 = -(\mathbf{v}'_2 - \mathbf{v}_2) = -\Delta\mathbf{v}_2. \quad (8.20)$$

When two hard disks collide, the force is exerted along the line connecting their centers, $\mathbf{r}_{12} = \mathbf{r}_1 - \mathbf{r}_2$. Hence, the components of the velocities parallel to \mathbf{r}_{12} are exchanged, and the perpendicular components of the velocities are unchanged. It is convenient to write the velocity of particles 1 and 2 as a vector sum of its components parallel and perpendicular to the unit vector $\hat{\mathbf{r}}_{12} = \mathbf{r}_{12}/|\mathbf{r}_{12}|$. We write the velocity of particle 1 as:

$$\mathbf{v}_1 = \mathbf{v}_{1,\parallel} + \mathbf{v}_{1,\perp}, \quad (8.21)$$

where $\mathbf{v}_{1,\parallel} = (\mathbf{v}_1 \cdot \hat{\mathbf{r}}_{12})\hat{\mathbf{r}}_{12}$,

$$\mathbf{v}'_{1,\parallel} = \mathbf{v}_{2,\parallel} \quad \mathbf{v}'_{2,\parallel} = \mathbf{v}_{1,\parallel} \quad (8.22a)$$

and

$$\mathbf{v}'_{1,\perp} = \mathbf{v}_{1,\perp} \quad \mathbf{v}'_{2,\perp} = \mathbf{v}_{2,\perp}. \quad (8.22b)$$

Hence, we can write \mathbf{v}'_1 as

$$\begin{aligned}\mathbf{v}'_1 &= \mathbf{v}'_{1,\parallel} + \mathbf{v}'_{1,\perp} \\ &= \mathbf{v}_{2,\parallel} + \mathbf{v}_{1,\perp} \\ &= \mathbf{v}_{2,\parallel} - \mathbf{v}_{1,\parallel} + \mathbf{v}_{1,\parallel} + \mathbf{v}_{1,\perp} \\ &= [(\mathbf{v}_2 - \mathbf{v}_1) \cdot \hat{\mathbf{r}}_{12}] \hat{\mathbf{r}}_{12} + \mathbf{v}_1.\end{aligned}\tag{8.23}$$

The change in the velocity of particle 1 at a collision is given by

$$\Delta \mathbf{v}_1 = \mathbf{v}'_1 - \mathbf{v}_1 = -[(\mathbf{v}_1 - \mathbf{v}_2) \cdot \hat{\mathbf{r}}_{12}] \hat{\mathbf{r}}_{12}\tag{8.24}$$

or

$$\Delta \mathbf{v}_1 = -\Delta \mathbf{v}_2 = \left(\frac{\mathbf{r}_{12} b_{12}}{\sigma^2} \right)_{\text{contact}},\tag{8.25}$$

where $b_{12} = \mathbf{v}_{12} \cdot \mathbf{r}_{12}$, $\mathbf{v}_{12} = \mathbf{v}_1 - \mathbf{v}_2$, and we have used the fact that $|\mathbf{r}_{12}| = \sigma$ at contact.

Problem 8.12. Velocity distribution of hard rods

Use (8.18) and (8.19) to show that $v'_1 = v_2$ and $v'_2 = v_1$ in one dimension, i.e., two colliding hard rods of equal mass exchange velocities. If you start a system of hard rods with velocities chosen from a uniform random distribution, will the velocity distribution approach the equilibrium Maxwell-Boltzmann distribution?

The most time consuming part of a hard disk dynamics program is computing the collision times of all pairs of particles. We now consider the criteria for a collision to occur. Consider disks 1 and 2 at positions \mathbf{r}_1 and \mathbf{r}_2 at $t = 0$. If they collide at a time t_{12} later, their centers will be separated by a distance σ :

$$|\mathbf{r}_1(t_{12}) - \mathbf{r}_2(t_{12})| = \sigma\tag{8.26}$$

During the time t_{12} , the disks move with constant velocities. Hence we have

$$\mathbf{r}_1(t_{12}) = \mathbf{r}_1(0) + \mathbf{v}_1(0) t_{12} \quad \text{and} \quad \mathbf{r}_2(t_{12}) = \mathbf{r}_2(0) + \mathbf{v}_2(0) t_{12}.\tag{8.27}$$

If we substitute (8.27) into (8.26), we find

$$[\mathbf{r}_{12} + \mathbf{v}_{12} t_{12}]^2 = \sigma^2\tag{8.28}$$

or

$$t_{12} = \frac{-\mathbf{v}_{12} \cdot \mathbf{r}_{12} \pm \sqrt{(\mathbf{v}_{12} \cdot \mathbf{r}_{12})^2 - v_{12}^2 (r_{12}^2 - \sigma^2)}}{v_{12}^2}.\tag{8.29}$$

Because $t_{12} > 0$ for a collision to occur, we see from (8.29) that the condition

$$\mathbf{v}_{12} \cdot \mathbf{r}_{12} < 0\tag{8.30}$$

must be satisfied. That is if $\mathbf{v}_{12} \cdot \mathbf{r}_{12} > 0$, the particles are moving away from each other and there is no possibility of a collision.

If the condition (8.30) is satisfied, then the discriminant in (8.29) must satisfy the condition

$$(\mathbf{v}_{12} \cdot \mathbf{r}_{12})^2 - v_{12}^2 (r_{12}^2 - \sigma^2) \geq 0. \quad (8.31)$$

If the condition (8.31) is satisfied, then the quadratic in (8.29) has two roots. The smaller root corresponds to the physically significant collision because the disks are impenetrable. Hence, the physically significant solution for the time of a collision t_{ij} for particles i and j is given by

$$t_{ij} = \frac{-b_{ij} - [b_{ij}^2 - v_{ij}^2 (r_{ij}^2 - \sigma^2)]^{\frac{1}{2}}}{v_{ij}^2}. \quad (8.32)$$

Problem 8.13. Calculation of collision times

Use (8.32) and SUB check_collision listed in Program hd to write a program that determines the collision times (if any) of the following pairs of particles. It would be a good idea to draw the trajectories to confirm your results. Consider the three cases: $\mathbf{r}_1 = (2, 1)$, $\mathbf{v}_1 = (-1, -2)$, $\mathbf{r}_2 = (1, 3)$, $\mathbf{v}_2 = (1, 1)$; $\mathbf{r}_1 = (4, 3)$, $\mathbf{v}_1 = (2, -3)$, $\mathbf{r}_2 = (3, 1)$, $\mathbf{v}_2 = (-1, -1)$; and $\mathbf{r}_1 = (4, 2)$, $\mathbf{v}_1 = (-2, \frac{1}{2})$, $\mathbf{r}_2 = (3, 1)$, $\mathbf{v}_2 = (-1, 1)$. As usual, choose units so that $\sigma = 1$.

The main thermodynamic quantity of interest for hard disks is the mean pressure P . Because the forces act only when two disks are in contact, we have to modify the form of (8.10). We write $\mathbf{F}_{ij}(t) = \mathbf{I}_{ij} \delta(t - t_c)$, where t_c is the time at which the collision occurs. This form of \mathbf{F}_{ij} implies the force is nonzero only when there is a collision between i and j . (The delta function $\delta(t)$ is infinite for $t = 0$ and is zero otherwise; $\delta(t)$ is defined by its use in an integral as shown in (8.33).) This form of the force yields

$$\int_0^t \mathbf{I}_{ij} \delta(t' - t_c) dt' = \mathbf{I}_{ij} = m \Delta \mathbf{v}_{ij}, \quad (8.33)$$

where we have used Newton's second law and assumed that a single collision has occurred during the time interval t . The quantity $\Delta \mathbf{v}_{ij} = \mathbf{v}'_i - \mathbf{v}_i - (\mathbf{v}'_j - \mathbf{v}_j)$. If we explicitly include the time average to account for all collisions during a time interval t , we can write (8.10) as

$$\begin{aligned} \frac{PV}{NkT} - 1 &= \frac{1}{dNkT} \frac{1}{t} \sum_{ij} \int_0^t \mathbf{r}_{ij} \cdot \mathbf{I}_{ij} \delta(t' - t_c) dt' \\ &= \frac{1}{dNkT} \frac{1}{t} \sum_{c_{ij}} m \Delta \mathbf{v}_{ij} \cdot \mathbf{r}_{ij} \end{aligned} \quad (8.34)$$

The sum in (8.34) is over all collisions c_{ij} between disks i and j in the time interval t ; \mathbf{r}_{ij} is the vector between the centers of the disks at the time of a collision; the magnitude of \mathbf{r}_{ij} in (8.34) is σ .

Our hard disk dynamics program implements the following steps. Given the initial condition in SUB initial, we find the collision times and the collision partners for all pairs of particles i and j . We then

1. locate the minimum collision time t_{\min} ;

2. advance all particles using a straight line trajectory until the collision occurs, that is, displace particle i by $\mathbf{v}_i t_{\min}$ and update the collision time;
3. compute the postcollision velocities of the colliding pair i and j ;
4. calculate any quantities of interest and accumulate data;
5. update the collision partners of the colliding pair i and j and any other particles that were to collide with either i or j if i and j had not collided first;
6. repeat steps 1–5 indefinitely.

This steps are implemented in Program `hd` which is listed in the following:

```

PROGRAM hd
! dynamics of system of hard disks
! program based in part on Fortran program of Allen and Tildesley
PUBLIC x(100),y(100),vx(100),vy(100)
PUBLIC collision_time(100),partner(100)
PUBLIC N,Lx,Ly,t,timebig
LIBRARY "csgraphics"
CALL initial(vsum,rho,area)
CALL set_up_windows(ncolor,#1,#2)
CALL kinetic_energy(ke,#1)
CALL show_positions(flag$,#2)
LET temperature = ke/N
LET flag$ = ""
LET collisions = 0           ! number of collisions
DO
  CALL minimum_collision_time(i,j,tij)
  ! move particles forward and reduce collision times by tij
  CALL move(tij)
  LET t = t + tij
  LET collisions = collisions + 1
  CALL show_positions(flag$,#2)
  CALL contact(i,j,virial)   ! compute collision dynamics
  LET vsum = vsum + virial
  CALL show_output(t,collisions,temperature,vsum,rho,area,#1)
  ! reset collision list for relevant particles
  CALL reset_list(i,j)
LOOP until flag$ = "stop"
CALL save_config(#2)
END

```

The colliding pair and the next collision time are found in SUB `minimum_collision_time`, and all particles are moved forward in SUB `move` until contact occurs. The collision dynamics of the colliding pair is computed in SUB `contact`, where the contribution to the pressure virial also is

found. In SUB `reset_list` we update the collision partners of the colliding pair (i and j) and any other particles that were to collide with i and j if i and j had not collided first.

In SUB `initial` we initialize various variables and most importantly, call SUB `uplist` and SUB `check_collision` to compute the collision time for each particle assuming that no other particles are present. Note that the i th element in the array `collision_time` contains the minimum collision time for particle i with all particles j such that $j > i$. The array `partner(i)` stores the particle f of the collision partner corresponding to this time with `partner(i)` always greater than i . The collision time for each particle is initially set to an arbitrarily large value, `timebig`, to account for the fact that at any given time, some particles have no collision partners.

```

SUB initial(vsum,rho,area)
  DECLARE PUBLIC x(),y(),vx(),vy()
  DECLARE PUBLIC N,Lx,Ly,t
  DECLARE PUBLIC collision_time(),partner(),timebig
  LET t = 0
  INPUT prompt "read file (f) or lattice start (l) = ": start$
  IF start$ = "f" or start$ = "F" then
    INPUT prompt "file name = ": file$
    OPEN #1: name file$, access input
    INPUT #1: N
    INPUT #1: Lx
    INPUT #1: Ly
    INPUT #1: heading$
    FOR i = 1 to N
      INPUT #1: x(i),y(i)
    NEXT i
    INPUT #1: heading$
    FOR i = 1 to N
      INPUT #1: vx(i),vy(i)
    NEXT i
    CLOSE #1
  ELSE IF start$ = "l" or start$ = "L" then
    RANDOMIZE
    INPUT prompt "N = ": N      ! choose N so that sqr(N) an integer
    INPUT prompt "Lx = ": Lx
    INPUT prompt "Ly = ": Ly
    INPUT prompt "vmax = ": vmax
    LET nx = sqr(N)
    LET ny = nx
    IF nx >= Lx or nx >= Ly then
      PRINT "box too small"
      STOP
    END IF
    LET ax = Lx/nx      ! "lattice" spacing
    LET ay = Ly/ny
    LET i = 0

```

```

    FOR col = 1 to nx
      FOR row = 1 to ny
        LET i = i + 1
        LET x(i) = (col - 0.5)*ax
        LET y(i) = (row - 0.5)*ay
        ! choose random positions and velocities
        LET vx(i) = (2*rnd - 1)*vmax
        LET vy(i) = (2*rnd - 1)*vmax
      NEXT row
    NEXT col
  END IF
  CLEAR
  CALL check_overlap          ! check if two disks overlap
  CALL check_momentum
  LET area = Lx*Ly
  LET rho = N/area
  LET timebig = 1.0e10
  LET vsum = 0                ! virial sum
  FOR i = 1 to N
    LET partner(i) = N
  NEXT i
  LET collision_time(N) = timebig
  ! set up initial collision lists
  FOR i = 1 to N
    CALL uplist(i)
  NEXT i
END SUB

SUB uplist(i)
  DECLARE PUBLIC N,collision_time(),timebig
  ! look for collisions with particles j > i
  IF i = N then EXIT SUB
  LET collision_time(i) = timebig
  FOR j = i + 1 to N
    CALL check_collision(i,j)
  NEXT j
END SUB

```

SUB `check_collision` uses the relations (8.30) and (8.32) to determine whether particles i and j will collide and if so, the time t_{ij} until their collision. We consider not only the nearest periodic image of j , but also the images of j in the adjoining cells, and determine the minimum collision time using all of these images. As shown in Figure 8.7, it is possible for i to collide with an image of j that is not the image closest to i . For dense systems the probability that j is in a more distant cell is very small and these distant cells are ignored in the program.

```

SUB check_collision(i,j)

```

```

DECLARE PUBLIC x(),y(),vx(),vy()
DECLARE PUBLIC Lx,Ly,collision_time(),partner()
! consider collisions between i and periodic images of j
FOR xcell = -1 to 1
  FOR ycell = -1 to 1
    LET dx = x(i) - x(j) + xcell*Lx
    LET dy = y(i) - y(j) + ycell*Ly
    LET dvx = vx(i) - vx(j)
    LET dvy = vy(i) - vy(j)
    LET bij = dx*dvx + dy*dvy
    IF bij < 0 then
      LET r2 = dx*dx + dy*dy
      LET v2 = dvx*dvx + dvy*dvy
      LET discr = bij*bij - v2*(r2 - 1)
      IF discr > 0 then
        LET tij = (-bij - sqr(discr))/v2
        IF tij < collision_time(i) then
          LET collision_time(i) = tij
          LET partner(i) = j
        END IF
      END IF
    END IF
  NEXT ycell
NEXT xcell
END SUB

```

The minimum collision time t_{ij} is found in SUB `minimum_collision_time` and all particles are moved forward by this time in SUB `move`.

```

SUB minimum_collision_time(i,j,tij)
DECLARE PUBLIC N,collision_time(),partner(),timebig
! locate minimum collision time
LET tij = timebig
FOR k = 1 to N
  IF collision_time(k) < tij then
    LET tij = collision_time(k)
    LET i = k
  END IF
NEXT k
LET j = partner(i)
END SUB

```

```

SUB move(tij)
DECLARE PUBLIC x(),y(),vx(),vy()
DECLARE PUBLIC N,collision_time()
DECLARE PUBLIC Lx,Ly

```

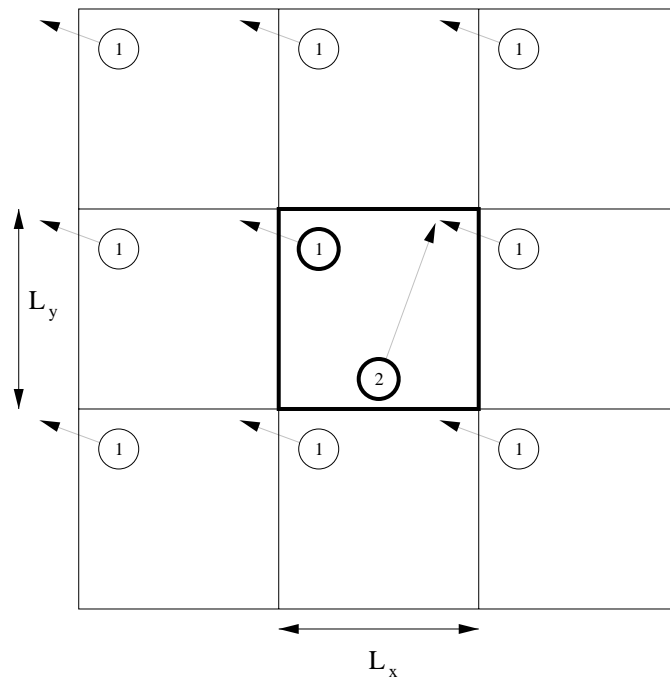


Figure 8.7: The positions and velocities of disks 1 and 2 are such that disk 1 collides with an image of disk 2 that is not the image closest to disk 1. The periodic images of disk 2 are not shown.

```

DECLARE DEF pbc
FOR k = 1 to N
  LET collision_time(k) = collision_time(k) - tij
  LET x(k) = x(k) + vx(k)*tij
  LET y(k) = y(k) + vy(k)*tij
  LET x(k) = pbc(x(k),Lx)
  LET y(k) = pbc(y(k),Ly)
NEXT k
END SUB

```

The function `pbc` allows for the possibility that a disk has moved further than the linear dimension of the central cell between a collision. We have written it as a separate function to emphasize its purpose.

```

DEF pbc(pos,L)
  LET pbc = mod(pos,L)
END DEF

```

The function `separation` is identical to the function listed in Program `md` and is not listed here.

The collision dynamics for the colliding particles i and j and the contribution of the collision to the virial are computed in SUB `contact`:

```

SUB contact(i,j,virial)
  DECLARE PUBLIC x(),y(),vx(),vy(),Lx,Ly
  DECLARE DEF separation
  ! compute collision dynamics for particles i and j at contact
  LET dx = separation(x(i) - x(j),Lx)
  LET dy = separation(y(i) - y(j),Ly)
  LET dvx = vx(i) - vx(j)
  LET dvy = vy(i) - vy(j)
  LET factor = dx*dvx + dy*dvy
  LET delvx = - factor*dx
  LET delvy = - factor*dy
  LET vx(i) = vx(i) + delvx
  LET vx(j) = vx(j) - delvx
  LET vy(i) = vy(i) + delvy
  LET vy(j) = vy(j) - delvy
  LET virial = delvx*dx + delvy*dy
END SUB

```

In SUB `reset_list` we find the new collision partners of particles i and j and those particles that were due to collide with i and j . Note that this update procedure must be done for particles whose labels are greater than i and j (SUB `uplist`) and for particles whose labels are less than i and j (SUB `downlist`).

```

SUB reset_list(i,j)
  DECLARE PUBLIC N,partner()
  ! reset collision list for relevant particles
  FOR k = 1 to N
    LET test = partner(k)
    IF k = i or test = i or k = j or test = j then
      CALL uplist(k)
    END IF
  NEXT k
  CALL downlist(i)
  CALL downlist(j)
END SUB

```

```

SUB downlist(j)
  DECLARE PUBLIC collision_time()
  ! look for collisions with particles i < j
  IF j = 1 then EXIT SUB
  FOR i = 1 to j - 1
    CALL check_collision(i,j)
  NEXT i
END SUB

```

SUB `check_momentum` and SUB `set_up_windows` are identical to the subroutines listed in Program `md` and are not listed here. SUB `save_config` is similar to the one listed in Program `md` except that the disks are moved so that none of the disks are in contact before the configuration is saved. Because the main loop of the program is not completed until two disks are in contact, this move is made to remove the possibility that two disks might appear to overlap due to floating point error.

```

SUB save_config(#2)
  DECLARE PUBLIC x(),y(),vx(),vy()
  DECLARE PUBLIC N,Lx,Ly
  WINDOW #2
  SET COLOR "black"
  ! move particles away from collision for final configuration
  CALL minimum_collision_time(i,j,tij)
  CALL move(0.5*tij)
  SET CURSOR 1,1
  INPUT prompt "name of saved configuration = ": file$
  OPEN #1: name file$, access output, create new
  PRINT #1: N
  PRINT #1: Lx
  PRINT #1: Ly
  PRINT #1: "x(i)","y(i)"
  FOR i = 1 to N
    PRINT #1, using "----.#####, ----.#####" : x(i),y(i)
  NEXT i
  PRINT #1: "vx(i)","vy(i)"
  FOR i = 1 to N
    PRINT #1, using "----.#####, ----.#####" : vx(i),vy(i)
  NEXT i
  CLOSE #1
END SUB

```

As discussed in Problem 8.14, an important check on the calculated trajectories of a hard disk system is that no two disks overlap. SUB `check_overlap` tests for this condition.

```

SUB check_overlap
  DECLARE PUBLIC x(),y()
  DECLARE PUBLIC N,Lx,Ly
  DECLARE DEF separation
  LET tol = 1.0e-4
  FOR i = 1 to N - 1
    FOR j = i + 1 to N
      LET dx = separation(x(i) - x(j),Lx)
      LET dy = separation(y(i) - y(j),Ly)
      LET r2 = dx*dx + dy*dy
      IF r2 < 1 then
        LET r = sqr(r2)

```

```

                IF (1 - r) > tol then
                    PRINT "particles ";i;" and ";j;"overlap"
                    STOP
                END IF
            END IF
        NEXT j
    NEXT i
END SUB

```

The remaining output subroutines are similar to those in Program md, but are listed below for completeness.

```

SUB headings(#1)
    WINDOW #1
    SET CURSOR 1,1
    PRINT using "#####": "collisions";
    PRINT,
    PRINT using "#####": "time";
    PRINT,
    PRINT using "#####.##": "<P>";
    PRINT,
    PRINT using "###.##": "T";
    PRINT,
END SUB

SUB show_output(t,collisions,temperature,vsum,rho,area,#1)
    WINDOW #1
    SET CURSOR 2,1
    SET COLOR "black/white"
    PRINT using "#####": collisions;
    PRINT,
    PRINT using "###.##": t;
    PRINT,
    LET mean_virial = vsum/(2*t)
    LET mean_pressure = rho*temperature + mean_virial/area
    PRINT using "###.##": mean_pressure;
END SUB

SUB show_positions(flag$,#2)
    DECLARE PUBLIC x(),y(),N,Lx,Ly
    IF key input then
        GET KEY k
        IF k = ord("r") then
            WINDOW #2
            CLEAR
            SET COLOR "black"

```

```

        BOX LINES 0,Lx,0,Ly
        LET flag$ = ""
    ELSEIF k = ord("s") then
        LET flag$ = "stop"
    ELSEIF k = ord("n") then
        LET flag$ = "no_show"
    END IF
END IF
IF flag$ <> "no_show" then
    SET COLOR "red"
    WINDOW #2
    FOR i = 1 to N
        PLOT x(i),y(i)
    NEXT i
END IF
END SUB

```

SUB `show_disks` can be substituted for SUB `show_positions` to represent the positions of the hard disks as circles rather than as points.

```

SUB show_disks(ncolor,flag$,#2)
    DECLARE PUBLIC x(),y(),N,Lx,Ly
    WINDOW #2
    IF key input then
        GET KEY k
        IF k = ord("r") then
            CLEAR
            BOX LINES 0,Lx,0,Ly
            LET flag$ = ""
        ELSEIF k = ord("s") then
            LET flag$ = "stop"
        ELSEIF k = ord("n") then
            LET flag$ = "no_show"
        END IF
    END IF
    IF flag$ <> "no_show" then
        LET ncolor = mod(ncolor,6) + 1
        IF ncolor = 1 then
            CLEAR
            BOX LINES 0,Lx,0,Ly
        END IF
        SET COLOR MIX(ncolor) rnd,rnd,rnd
        SET COLOR ncolor
        FOR i = 1 to N
            BOX CIRCLE x(i)-0.5,x(i)+0.5,y(i)-0.5,y(i)+0.5
        NEXT i
    END IF
END SUB

```

```

END IF
END SUB

```

Finally, we list SUB `kinetic_energy` which is needed to compute the kinetic temperature:

```

SUB kinetic_energy(ke,#1)
  DECLARE PUBLIC vx(),vy()
  DECLARE PUBLIC N
  WINDOW #1
  LET ke = 0
  FOR i = 1 to N
    LET ke = ke + vx(i)*vx(i) + vy(i)*vy(i)
  NEXT i
  LET ke = 0.5*ke
  SET CURSOR 2,1
  PRINT,,
  PRINT using "##.###": ke/N;
  PRINT,
END SUB

```

Problem 8.14. Initial tests of Program `hd`

1. Because even a small error in computing the trajectories of the disks will eventually lead to their overlap and hence to a fatal error, it is necessary to test Program `hd` carefully. For simplicity, start from a lattice configuration. The most important test of the program is to monitor the computed positions of the hard disks at regular intervals for overlaps. If the distance between the centers of any two hard disks is less than unity (distances are measured in units of σ), there must be a serious error in the program. To check for the overlap of hard disks, include SUB `overlap` in the main loop of Program `hd` while you are testing the program.
2. The temperature for a system of hard disks is constant and can be defined as in (8.7). Why does the temperature not fluctuate as it does for a system of particles interacting with a continuous potential? The constancy of the temperature can be used as another check on your program. What is the effect of increasing all the velocities by a factor of two? What is the natural unit of time? Explain why the state of the system is determined only by the density and not by the temperature.
3. Use Program `hd` to generate equilibrium configurations of a system of $N = 16$ disks in a square cell of linear dimension $L = 6$. Suppose that at $t = 0$, the constraint that $0 \leq x \leq 6$ is removed, and the disks are allowed to move in a rectangular cell with $L_x = 12$ and $L_y = 6$. Does the system become more or less random? What is the qualitative nature of the time dependence of $n(t)$, the number of disks on the left half of the cell?
4. Modify your program so that averages are not computed until the system is in equilibrium. For simplicity, use the initial positions in the DATA statements in Program `md` as the initial condition. Compute the virial (8.34) and make a rough estimate of the error in your determination of the mean pressure due to statistical fluctuations.

5. Modify your program so that you can compute the velocity and speed distributions and verify that the computed distributions have the desired forms.

Problem 8.15. Static properties of hard disks

1. As we have seen in Section 8.7, a very time consuming part of the simulation is equilibrating the system from an arbitrary initial configuration. One way to obtain a set of initial positions is to add the hard disks sequentially with random positions and reject an additional hard disk if it overlaps any disks already present. Although this method is very inefficient at high densities, try it so that you will have a better idea of how difficult it is to obtain a high density configuration in this way. A much better method is to place the disks on the sites of a lattice.
2. We first consider the dependence of the mean pressure P on the density ρ . Is P a monotonically increasing function of ρ ? Is a system of hard disks always a fluid or is there a fluid to solid transition at higher densities? We will not be able to find definitive answers to these questions for $N = 16$. However, many simulations in the 1960's and 70's were done for systems of $N = 108$ hard disks and the largest simulations were for several hundred particles.
3. Compute the radial distribution function $g(r)$ for the same densities as you considered for the Lennard-Jones interaction. Compare the qualitative behavior of $g(r)$ for the two interactions. On the basis of your results, which part of the Lennard-Jones interaction plays the dominant role in determining the structure of a dense Lennard-Jones liquid?
4. The largest number of hard disks that can be placed into a fixed volume defines the maximum density. What is the maximum density if the disks are placed on a square lattice? What is the maximum density if the disks are placed on a triangular lattice? Suppose that the initial condition is chosen to be a square lattice with $N = 100$ and $L = 11$ so that each particle has four nearest neighbors initially. What is the qualitative nature of the system after several hundred collisions have occurred? Do most particles still have four nearest neighbors or are there regions where most particles have six neighbors?

In Problem 8.16 we consider two physical quantities associated with the dynamics of a system of hard disks, namely the mean free time and the mean free path, quantities that are discussed in texts on kinetic theory (cf. Reif).

Problem 8.16. Mean free path and collision time

1. Program `hd` provides the information needed to determine the mean free time t_c , i.e., the average time a particle travels between collisions. For example, suppose we know that 40 collisions occurred in a time $t = 2.5$ for a system of $N = 16$ disks. Because two particles are involved in each collision, there was an average of $80/16$ collisions per particle. Hence $t_c = 2.5/(80/16) = 0.5$. Write a subroutine to compute t_c and determine t_c as a function of ρ .
2. Write a subroutine to determine the distribution of times between collisions. What is the qualitative form of the distribution? How does the width of this distribution depend on ρ ?

3. The mean free path ℓ is the mean distance a particle travels between collisions. Is ℓ simply related to t_c by the relation $\ell = \bar{v}t_c$, where $\bar{v} = \sqrt{\overline{v^2}}$? Write a subroutine to compute the mean free path of the particles. Note that the displacement of particle i during the time t is $v_i t$, where v_i is the speed of particle i .

8.10 Dynamical Properties

The mean free time and the mean free path are well defined for hard disks for which the meaning of a collision is clear. As discussed in texts on kinetic theory (cf. Reif), both quantities are related to the transport properties of a dilute gas. However, the concept of a collision is not well-defined for systems with a continuous interaction such as the Lennard-Jones potential. In the following, we take a more general approach to the dynamics of a many body system and discuss how the transport of particles in a system near equilibrium is related to the equilibrium properties of the system.

Suppose that we tag a certain fraction of the particles in our system (hem blue), and let $n(\mathbf{r}, t)$ be the mean number density of the tagged particles, that is, $n(\mathbf{r}, t)d\mathbf{r}$ is the mean number of particles with positions in the region $d\mathbf{r}$ about \mathbf{r} at time t . In equilibrium, we expect that the tagged particles would be distributed uniformly and hence $n(\mathbf{r}, t)$ would be independent of \mathbf{r} and t . Suppose however, that we start with a nonuniform distribution of tagged particles and ask how $n(\mathbf{r}, t)$ changes with \mathbf{r} and t to make the density of tagged particles more uniform. We expect that \mathbf{J} , the flux of tagged particles, is proportional to the density gradient of the tagged particles. In one dimension we write

$$J_x = -D \frac{\partial n(x, t)}{\partial x}. \quad (8.35a)$$

More generally, we have that

$$\mathbf{J} = -D \nabla n(\mathbf{r}, t). \quad (8.35b)$$

The empirical relation (8.35) is known as Fick's law and expresses the fact that the flow of tagged particles acts to equalize the density. The quantity D in (8.35) is known as the **self-diffusion coefficient**. If we combine (8.35) with the statement that the number of tagged particles is conserved:

$$\frac{\partial n(\mathbf{r}, t)}{\partial t} + \nabla \cdot \mathbf{J}(\mathbf{r}, t) = 0, \quad (8.36)$$

we obtain the diffusion equation:

$$\frac{\partial n(\mathbf{r}, t)}{\partial t} = D \nabla^2 n(\mathbf{r}, t). \quad (8.37)$$

As the above discussion implies, we usually think of the transport of particles (and the transport of energy and momentum and other quantities) in the context of nonequilibrium situations. However, as we have already seen in our simulations, the particles in an equilibrium system are in continuous motion and hence continuously create local density fluctuations. We expect that these spontaneous density fluctuations behave in the same way as the density fluctuations that

are created by weak external perturbations. Hence, we also expect (8.37) to apply to a system in equilibrium.

Consider the trajectory of a particular particle, e.g., particle 1, in equilibrium. At some arbitrarily chosen time $t = 0$, its position is $\mathbf{r}_1(0)$. At a later time t , its displacement is $\mathbf{r}_1(t) - \mathbf{r}_1(0)$. If there were no net force on the particle during this time interval, then $\mathbf{r}_1(t) - \mathbf{r}_1(0)$ would increase linearly with t . However, a particle in a fluid undergoes many collisions and on the average its net displacement would be zero. A more interesting quantity is the mean square displacement defined as

$$\overline{R_1^2(t)} = \overline{[\mathbf{r}_1(t) - \mathbf{r}_1(0)]^2}. \quad (8.38)$$

The average in (8.38) is over all possible choices of the time origin. If the system is in equilibrium, the choice of $t = 0$ is arbitrary, and $\overline{R_1^2(t)}$ depends only on the time difference t . We have seen in Appendix 7A that the diffusion equation (8.37) implies that the t dependence of $\overline{R^2(t)}$ is given by

$$\overline{R^2(t)} = 2dDt, \quad (t \rightarrow \infty) \quad (8.39)$$

where d is the spatial dimension. We have omitted the subscript because the average behavior of all the particles is the same.

The relation (8.39) relates the macroscopic transport coefficient D to a microscopic quantity, $\overline{R^2(t)}$, and gives us a straightforward way of computing D for an equilibrium system. The easiest way of computing $\overline{R^2(t)}$ is to write the position of a particle at regular time intervals into a file. We later can use a separate program to read the data file and compute $\overline{R^2(t)}$. Of course, we would find much better results if we average over all particles.

To understand the procedure for computing $\overline{R^2(t)}$, we consider a simple example. Suppose that the position of a particle in a one-dimensional system is given by $x(t = 0) = 1.65$, $x(t = 1) = 1.62$, $x(t = 2) = 1.84$, and $x(t = 3) = 2.22$. If we average over all possible time origins, we obtain

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

Note that there are fewer combinations of the positions as the time difference increases. Program R2, listed in the following, reads a data file consisting of the x and y coordinates of a particle and computes $\overline{R^2(t)}$ by averaging over all possible choices of the time origin.

PROGRAM R2

```
! compute mean square displacement of one particle
! by averaging over all possible origins
```

```

DIM x(1000),y(1000),R2cum(20),norm(20)
CALL initial(Lx,Ly,R2cum(),ndiff)
CALL read_data(x(),y(),ndata)
CALL displacements(x(),y(),Lx,Ly,R2cum(),norm(),ndata,ndiff)
CALL normalization(ndiff,R2cum(),norm())
END

SUB initial(Lx,Ly,R2cum(),ndiff)
  LET Lx = 5
  LET Ly = 5
  LET ndiff = 10           ! maximum time difference
  FOR idiff = 1 to ndiff
    LET R2cum(idiff) = 0
  NEXT idiff
END SUB

SUB read_data(x(),y(),ndata)
  ! read file for position of particle at regular intervals
  OPEN #1: name "xy.dat",access input
  LET t = 0
  DO while more #1
    LET t = t + 1
    INPUT #1: x(t),y(t)
  LOOP
  CLOSE #1
  LET ndata = t           ! # of data points
END SUB

SUB displacements(x(),y(),Lx,Ly,R2cum(),norm(),ndata,ndiff)
  DECLARE DEF separation           ! function same as in Program md
  FOR idiff = 1 to ndiff
    FOR i = 1 to ndata - idiff
      LET dx = separation(x(i+idiff) - x(i),Lx)
      LET dy = separation(y(i+idiff) - y(i),Lx)
      LET R2cum(idiff) = R2cum(idiff) + dx*dx + dy*dy
      LET norm(idiff) = norm(idiff) + 1
    NEXT i
  NEXT idiff
END SUB

SUB normalization(ndiff,R2cum(),norm())
  PRINT "time difference"," <R2>"
  PRINT
  FOR idiff = 1 to ndiff
    IF R2cum(idiff) > 0 then
      LET R2bar = R2cum(idiff)/norm(idiff)
    END IF
  NEXT idiff
END SUB

```

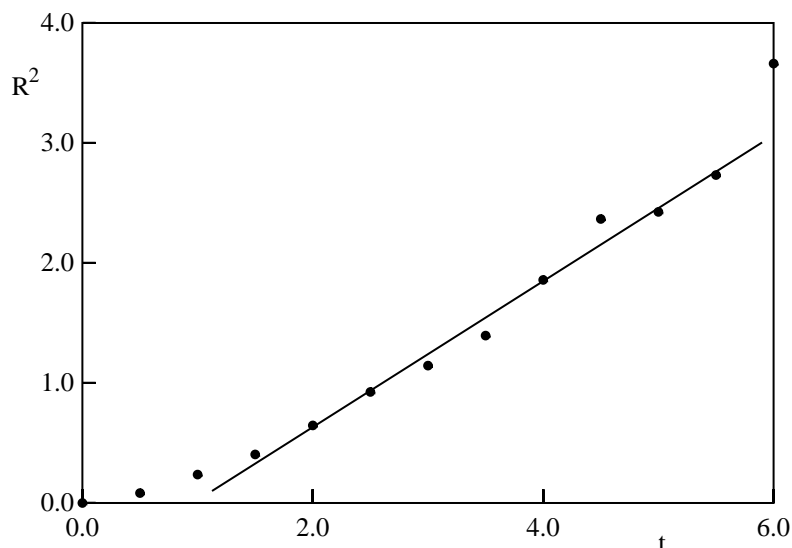


Figure 8.8: The time dependence of the mean square displacement $\overline{R^2(t)}$ for one particle in a two-dimensional Lennard-Jones system with $N = 16$, $L = 5$, and $E = 5.8115$. The position of a particle was saved at intervals of 0.5. Much better results can be obtained by averaging over all 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.

```

        PRINT idiff,R2bar
    END IF
NEXT idiff
END SUB

```

We show our results for $\overline{R^2(t)}$ for a system of Lennard-Jones particles in Figure 8.8. Note that $\overline{R^2(t)}$ 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 8.17 we use Program R2 to compute the self-diffusion coefficient. An alternative way of computing D is discussed in Project 8.19.

Problem 8.17. The self-diffusion coefficient

1. Use Program md or Program hd and visually follow the motion of a particular particle by “tagging” it, e.g., by drawing its path with a different color. Describe its motion qualitatively.
2. Modify Program md or Program hd so that the coordinates of a particular particle are saved at regular intervals. The desired time interval needs to be determined empirically. If we save the coordinates too often, the data file will be too large, and we will waste time writing the data to a file. If we do not save the positions often enough, we 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 a factor of ten greater than Δt . A good first guess is to choose the time interval to be the order of 10–50 time steps. The easiest procedure for hard disks is to save the positions at intervals measured in terms of the number of collisions. If we average over a sufficient number of collisions, we can find the relation between the elapsed time and the number of collisions.

3. Compute $\overline{R^2(t)}$ for conditions that correspond to a dense fluid. Does $\overline{R^2(t)}$ increase as t^2 as for a free particle or more slowly? Does $\overline{R^2(t)}$ increase linearly with t for longer times? What is the maximum value of $\overline{R^2(t)}$ that you find? Why does the use of periodic boundary conditions imply that there is an upper bound to the maximum time difference t we can consider when computing $\overline{R^2(t)}$?
4. Use the relation (8.39) to estimate the magnitude of D from the slope of $\overline{R^2(t)}$. Obtain D for several different temperatures and densities. (A careful study of $\overline{R^2(t)}$ for much larger systems and much longer times would show that $\overline{R^2(t)}$ 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 if t is sufficiently large. However, we will not be able to observe the effects of this logarithmic term, and we can interpret our results for $\overline{R^2(t)}$ in terms of an “effective” diffusion coefficient. No such problem exists for three dimensions.)
5. Compute $\overline{R^2(t)}$ for an equilibrium configuration corresponding to a harmonic solid. What is the qualitative behavior of $\overline{R^2(t)}$?
6. Compute $\overline{R^2(t)}$ for an equilibrium configuration corresponding to a dilute gas. Is $\overline{R^2(t)}$ proportional to t for small times? Can we consider the particles to diffuse over short time intervals?

Another physically important single particle property is the **velocity autocorrelation function** $\psi(t)$. Suppose that particle i has velocity \mathbf{v}_i at time t_1 . If there were no net force on particle i , its velocity would remain constant. However, the interactions with other particles in the fluid will change the particle’s velocity, and we expect that after several collisions, its velocity will not be strongly correlated with its velocity at an earlier time. We define $\psi(t)$ as

$$\psi(t) = \frac{1}{v_0^2} \overline{\mathbf{v}_i(t_2) \cdot \mathbf{v}_i(t_1)}, \quad (8.40)$$

where $v_0^2 = \overline{\mathbf{v}_i(0) \cdot \mathbf{v}_i(0)} = dkT/m$ and $t = t_2 - t_1$. As in our discussion of the mean square displacement, the average in (8.40) is over all possible time origins. We have defined $\psi(t)$ such that $\psi(t = 0) = 1$. For large time differences $t_2 - t_1$, we expect $\mathbf{v}_i(t_2)$ to be independent of $\mathbf{v}_i(t_1)$, and hence $\psi(t) \rightarrow 0$ for $t \rightarrow \infty$. (Note that we have implicitly assumed that $\overline{\mathbf{v}_i(t)} = 0$.) It can be shown that the self-diffusion coefficient defined by (8.39) can be related to an integral of $\psi(t)$:

$$D = v_0^2 \int_0^\infty \psi(t) dt. \quad (8.41)$$

Other transport coefficients such as the shear viscosity and the thermal conductivity can also be expressed as a time integral over a corresponding autocorrelation function. The qualitative properties of the velocity autocorrelation function are explored in Problem 8.18.

**Problem 8.18.* The velocity autocorrelation function

1. Modify your molecular dynamics program so that the velocity of a particular particle is saved to a file at regular time intervals. Then modify **Program R2** so that you can compute $\psi(t)$. The following code might be useful.

```
FOR idiff = 1 to ndiff
  FOR i = 1 to nmax - idiff
    LET psi(idiff) = psi(idiff) + vx(i + idiff)*vx(i)
    LET psi(idiff) = psi(idiff) + vy(i + idiff)*vy(i)
    LET norm(idiff) = norm(idiff) + 1
  NEXT i
NEXT idiff
```

Compute $\psi(t)$ for the same equilibrium configurations as in Problem 8.17c. Plot $\psi(t)$ versus t and describe its qualitative behavior. Estimate D from the relation (8.41). (To estimate the integral of $\psi(t)$, add your results for $\psi(t)$ at the different values of t and multiply the sum by the time difference between successive values of t .) How does your result for D compare to the determination using (8.39)?

2. Assume that $\psi(t)$ satisfies the form $\psi(t) = e^{-t/t_r}$ for all t . Substitute this form for $\psi(t)$ into (8.41) and determine the relationship between D and the **relaxation time** t_r . Plot the natural logarithm of $\psi(t)$ versus t and estimate t_r from the linear behavior of $\ln \psi(t)$. (At very long times, $\psi(t)$ exhibits slower than exponential decay. This “long-time tail” is due to hydrodynamic effects.) Use your derived relationship between D and t_r to find D . Compare your estimates for D found from the slope of $\overline{R^2(t)}$, the relation (8.41), and the estimate of t_r . Are these estimates consistent?
3. Increase the density by 50% and compute $\psi(t)$. What is the qualitative behavior of $\psi(t)$? What is the implication of the fact that $\psi(t)$ becomes negative after a relatively short time?
4. Compute $\psi(t)$ for an equilibrium solid. Plot $\psi(t)$ versus t and describe its qualitative behavior. Explain your results in terms of the oscillatory motion of the particles about their lattice sites.
5. Contrast the behavior of the mean square displacement, the velocity autocorrelation function, and the radial distribution function in the solid and fluid phases and explain how these quantities can be used to indicate the nature of the phase.
6. Modify your program so that $\overline{R^2(t)}$ and $\psi(t)$ are averaged over all particles.

8.11 Extensions

The primary goals of this chapter have been to introduce the method of molecular dynamics and some of the concepts of statistical mechanics and kinetic theory. Although we found that simulations of systems as small as sixteen particles show some of the qualitative properties of macroscopic systems, we would need to simulate larger systems to make quantitative conclusions.

How do we know if the size of our system is sufficiently large to yield quantitative results that are independent of N ? The straightforward answer is to repeat the simulation for larger N . Fortunately, most simulations of equilibrium systems with simple interactions require only several hundred to several thousand particles for reliable results. How do we know if our runs are long enough to give statistically meaningful averages? The simple answer is to run longer and see if the averages change significantly.

In general, the most time consuming parts of a molecular dynamics simulation are generating an appropriate initial configuration and doing the bookkeeping necessary for the force and energy calculations. If the force is sufficiently short range, there are a number of ways to reduce the equilibration time. For example, suppose we want to simulate a system of 864 particles in three dimensions. We first can simulate a system of 108 particles and allow the small system to come to equilibrium at the desired temperature. After equilibrium has been established, the small system can be replicated twice in each direction to generate the desired system of 864 particles. All of the velocities are reassigned at random using the Maxwell-Boltzmann distribution. Equilibration of the new system usually is established quickly.

The computer time required for our simple molecular dynamics program is order N^2 for each time step. The reason for this quadratic dependence on N is that the energy and force calculations 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 at any given time, most pairs of particles are separated by a distance much greater than the effective range r_c of the interparticle interaction ($r_c \approx 2.5\sigma$ for the Lennard-Jones potential). Hence the calculation of the force and the energy requires the consideration of 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 method is to divide the box into small cells and to 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_l , where r_l is chosen to be slightly greater than r_c so that the neighbor list can be used for several time steps before it is updated again. Both the cell method and the neighbor list method do not become efficient until N is approximately a few hundred.

So far we have discussed molecular dynamics simulations at fixed energy, volume, and number of particles. In laboratory systems we usually keep the temperature rather than the energy fixed, and frequently consider systems at fixed pressure rather than at fixed volume. It is possible to do molecular dynamics simulations at constant temperature and/or pressure. It also is possible 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 technical advances, there is much more to learn about the properties of the system by doing averages over the trajectories. For example, how are transport properties such as the viscosity and the thermal conductivity related to the trajectories? We also have 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 can be simply defined as a time average over some function of the phase space coordinates of the particles. The entropy is an example of such a quantity (but see Ma). However, changes in the entropy can be computed by using thermodynamic integration or a test particle method (see references).

There is another fundamental limitation of molecular dynamics, namely the **multiple time scale problem**. We know that 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 if we add an interstitial particle or create a vacancy, we would have to run for millions of small time steps for the vacancy to move several interparticle distances. Although this particular problem can be overcome by using a faster computer, there are many problems for which no imaginable supercomputer would be sufficient. One of the biggest challenges of present interest is the **protein folding problem**. The biological function of a protein is determined by its three-dimensional structure which is encoded by the sequence of amino acids in the protein. At present, we know little about how the protein forms its three-dimensional structure. Such formidable computational challenges remind us that we cannot simply put a problem on a computer and let the computer tell us the answer. In particular, molecular dynamics methods need to be complemented by other simulation methods, especially Monte Carlo methods (see Chapters ?? and ??).

Now that we are familiar with the method of molecular dynamics, we briefly discuss its historical role in aiding our present understanding of simple equilibrium liquids. Simulations of systems of hard disks and hard spheres have shown that the structure of these systems does not differ significantly from the structure of systems with more complicated interactions. Given this insight, our present theories of liquids are based on the use of the hard sphere (disk) system as a reference system; the differences between the hard sphere interaction and the more complicated interaction of interest are treated as a perturbation about this reference system.

The emphasis in current applications of molecular dynamics is shifting from the studies of simple equilibrium fluids to studies of more complex fluids and studies of nonequilibrium systems. For example, how does a solid form when the temperature of a liquid is lowered quickly? How does a crack propagate in a brittle solid? What is the nature of the glass transition? Molecular dynamics and related methods will play an important role in aiding our understanding of these and many other problems.

8.12 Projects

Many of the pioneering applications of molecular dynamics were done on relatively small systems. It is interesting to peruse the research literature of the past three decades and to see how much physical insight was obtained from these simulations. Many research-level problems can be generated by first reproducing previously published work and then extending the work to larger systems or longer run times to obtain better statistics. An interesting project based on recent research is suggested in the following. Some related projects are discussed in Section ??.

Project 8.19. Single particle fluctuation metric

As we discussed briefly in Section 8.7, the quasi-ergodic hypothesis assumes that time averages and ensemble averages are identical for a system in thermodynamic equilibrium. The assumption is that if we run a molecular dynamics simulation for a sufficiently long time, then the dynamical trajectory will fill the accessible phase space.

One way to confirm the quasi-ergodic hypothesis is to compute an ensemble average by simulating many independent copies of the system of interest using different initial configurations.

Another way is to simulate a very large system and compare the behavior of different parts. A more direct and computationally efficient measure of the ergodicity has been proposed by Thirumalai and Mountain. This measure is called the fluctuation metric and is based on a comparison of the time averaged quantity $\overline{f_i(t)}$ of f_i for particle i to its average for all other particles. Effectively, we take the ensemble average of f over all particles, rather than over different parts of the system. If the system is ergodic, then all particles see the same average environment, and the time average $\overline{f_i(t)}$ for each particle will be the same if t is sufficiently long. Note that $\overline{f_i(t)}$ is the average of the quantity f_i over the time interval t and not the value of f_i at time t . The time average of f_i is defined as

$$\overline{f_i(t)} = \frac{1}{t} \int_0^t f(t') dt', \quad (8.42)$$

and the average of $\overline{f_i(t)}$ over all particles is given by

$$\langle f(t) \rangle = \frac{1}{N} \sum_{i=1}^N \overline{f_i(t)}. \quad (8.43)$$

One of the physical quantities of interest is the energy of a particle, e_i , defined as

$$e_i = \frac{p_i^2}{2m_i} + \frac{1}{2} \sum_{i \neq j} u(r_{ij}). \quad (8.44)$$

The factor of 1/2 is included in the potential energy term in (8.44) because the interaction energy is shared between pairs of particles. The above considerations lead us to define the energy fluctuation metric, $\Omega_e(t)$, as

$$\Omega_e(t) = \frac{1}{N} \sum_{i=1}^N \left[\overline{e_i(t)} - \langle e(t) \rangle \right]^2. \quad (8.45)$$

1. Compute $\Omega_e(t)$ for a system of Lennard-Jones particles at a relatively high temperature. Determine $e_i(t)$ at time intervals of 0.5 or less and average Ω_e over as many time origins as possible. If the system is displaying behavior that is expected of an ergodic system over the time interval t , it can be shown that $\Omega_e(t)$ decreases as $1/t$. Do you find $1/t$ behavior for relatively short times? Nonergodic behavior might be found by rapidly reducing the kinetic energy (a temperature quench) and obtaining an amorphous solid or glass rather than a crystalline solid. However, it would be necessary to consider three-dimensional rather than two-dimensional systems because the latter nucleate to a crystalline solid very quickly.
2. Another quantity of interest is the velocity fluctuation metric Ω_v :

$$\Omega_v(t) = \frac{1}{dN} \sum_{i=1}^N \left[\overline{\mathbf{v}_i(t)} - \langle \mathbf{v}(t) \rangle \right]^2. \quad (8.46)$$

The factor of $1/d$ in (8.46) is included because the velocity is a vector with d components. If we choose the total momentum of the system to be zero, then $\langle \mathbf{v}(t) \rangle = 0$, and we can write

(8.46) as

$$\Omega_v(t) = \frac{1}{dN} \sum_{i=1}^N \overline{\mathbf{v}_i(t) \cdot \mathbf{v}_i(t)}. \quad (8.47)$$

We now show that the t dependence of $\Omega_v(t)$ is not a good indicator of ergodicity, but can be used to determine the diffusion coefficient D . We write

$$\overline{\mathbf{v}_i(t)} = \frac{1}{t} \int_0^t \mathbf{v}_i(t') dt' = \frac{1}{t} [\mathbf{r}_i(t) - \mathbf{r}_i(0)]. \quad (8.48)$$

If we substitute (8.48) into (8.47), we can express the velocity fluctuation metric in terms of the mean square displacement:

$$\Omega_v(t) = \frac{1}{dNt^2} \sum_{i=1}^N [\mathbf{r}_i(t) - \mathbf{r}_i(0)]^2 = \frac{\langle R^2(t) \rangle}{dt^2}. \quad (8.49)$$

The average in (8.49) is over all particles. If the particles are diffusing during the time interval t , then $\langle R^2(t) \rangle = 2dDt$, and

$$\Omega_v(t) = 2D/t. \quad (8.50)$$

From (8.50) we see that $\Omega_v(t)$ goes to zero as $1/t$ as claimed in part (a). However, if the particles are localized (as in a crystalline solid and a glass), then $\langle R^2 \rangle$ is bounded for all t , and $\Omega_v(t) \sim 1/t^2$. Because a crystalline solid is ergodic and a glass is not, the velocity fluctuation metric is not a good measure of the lack of ergodicity. Use the t dependence of $\Omega_v(t)$ in (8.50) to determine D for the same configurations as in Problem 8.17. Note that the determination of D from Ω_v does not require a correction for the use of periodic boundary conditions.

References and Suggestions for Further Reading

- Farid F. Abraham, "Computational statistical mechanics: methodology, applications and super-computing," *Adv. Phys.* **35**, 1 (1986). The author discusses both molecular dynamics and Monte Carlo techniques.
- M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Clarendon Press (1987). See Chapter 7 for a discussion of the test particle method.
- R. P. Bonomo and F. Riggi, "The evolution of the speed distribution for a two-dimensional ideal gas: A computer simulation," *Am. J. Phys.* **52**, 54 (1984). The authors consider a system of hard disks and show that the system always evolves toward the Maxwell-Boltzmann distribution.
- J. P. Boon and S. Yip, *Molecular Hydrodynamics*, Dover (1991). Their discussion of transport properties is an excellent supplement to our brief discussion.

- Giovanni Ciccotti and William G. Hoover, editors, *Molecular-Dynamics Simulation of Statistical-Mechanics Systems*, North-Holland (1986).
- Giovanni Ciccotti, Daan Frenkel, and Ian R. McDonald, editors, *Simulation of Liquids and Solids*, North-Holland (1987). A collection of reprints on the simulation of many body systems. Of particular interest are B. J. Alder and T. E. Wainwright, "Phase transition in elastic disks," *Phys. Rev.* **127**, 359 (1962) and earlier papers by the same authors; A. Rahman, "Correlations in the motion of atoms in liquid argon," *Phys. Rev.* **136**, A405 (1964), the first application of molecular dynamics to systems with continuous potentials; and Loup Verlet, "Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules," *Phys. Rev.* **159**, 98 (1967).
- J. M. Haile, *Molecular Dynamics Simulation*, John Wiley & Sons (1992). A derivation of the mean pressure using periodic boundary conditions is given in Appendix B.
- Jean Pierre Hansen and Ian R. McDonald, *Theory of Simple Liquids*, second edition, Academic Press (1986). An excellent reference that derives most of the theoretical results used in this chapter.
- R. M. Hockney and J. W. Eastwood, *Computer Simulation Using Particles*, Adam Hilger (1988).
- W. G. Hoover, *Molecular Dynamics*, Springer-Verlag (1986) and W. G. Hoover, *Computational Statistical Mechanics*, Elsevier (1991).
- J. Kushick and B. J. Berne, "Molecular dynamics methods: continuous potentials" in *Statistical Mechanics Part B: Time-Dependent Processes*, Bruce J. Berne, editor, Plenum Press (1977). Also see the article by Jerome J. Erpenbeck and William Wood on "Molecular dynamics techniques for hard-core systems" in the same volume.
- Shang-keng Ma, "Calculation of entropy from data of motion," *J. Stat. Phys.* **26**, 221, (1981). See also Chapter 25 of Ma's graduate level text, *Statistical Mechanics*, World Scientific (1985). Ma discusses a novel approach for computing the entropy directly from the trajectories. Note that the coincidence rate in Ma's approach is related to the recurrence time for a finite system to return to an arbitrarily small neighborhood of almost any given initial state.
- S. Ranganathan, G. S. Dubey, and K. N. Pathak, "Molecular-dynamics study of two-dimensional Lennard-Jones fluids," *Phys. Rev. A* **45**, 5793 (1992). Two-dimensional systems are of interest because they are simpler theoretically and computationally and are related to single layer films.
- Dennis Rapaport, *The Art of Molecular Dynamics Simulation*, Cambridge University Press (1995).
- John R. Ray and H. W. Graben, "Direct calculation of fluctuation formulae in the microcanonical ensemble," *Mol. Phys.* **43**, 1293 (1981).
- F. Reif, *Fundamentals of Statistical and Thermal Physics*, McGraw-Hill (1965.) An intermediate level text on statistical physics with a more thorough discussion of kinetic theory than found in most undergraduate texts. *Statistical Physics*, Vol. 5 of the Berkeley Physics Course, McGraw-Hill (1965) by the same author was one of the first texts to use computer simulations to illustrate the approach of macroscopic systems to equilibrium.

- Marco Ronchetti and Gianni Jacucci, editors, *Simulation Approach to Solids*, Kluwer Academic Publishers (1990). Another collection of reprints.
- R. M. Sperandio Mineo and R. Madonia, "The equation of state of a hard-particle system: a model experiment on a microcomputer," *Eur. J. Phys.* **7**, 124 (1986).
- D. Thirumalai and Raymond D. Mountain, "Ergodic convergence properties of supercooled liquids and glasses," *Phys. Rev. A* **42**, 4574 (1990).
- James H. Williams and Glenn Joyce, "Equilibrium properties of a one-dimensional kinetic system," *J. Chem. Phys.* **59**, 741 (1973). Simulations in one dimension are even easier than in two.