Chapter 7

Random Processes

©2005 by Harvey Gould, Jan Tobochnik, and Wolfgang Christian
3 March 2005

Random processes are introduced in the context of several simple physical systems, including random walks on a lattice and polymers.

7.1 Order to Disorder

In Chapter 6 we saw several examples of how under certain conditions, the behavior of a nonlinear deterministic system can be so complicated that it appears to be random. In this chapter we will see some examples of how chance can generate statistically predictable outcomes. For example, we know that if we bet often on the outcome of a game for which the probability of winning is less than 50%, we will lose money eventually.

We first discuss an example that illustrates the tendency of systems of many particles to evolve toward a well defined state. Imagine a closed box that is divided into two parts of equal volume (see Figure 7.1). The left half contains a gas of N identical particles and the right half is empty. We then make a small hole in the partition between the two halves. What happens? We know that after some time, the average number of particles in each half of the box will be $N/2$, and we say that the system has reached equilibrium.

How can we simulate this process? One way is to give each particle an initial velocity and position and adopt a deterministic model of the motion of the particles. For example, we could assume that each particle moves in a straight line until it hits a wall of the box or another particle and undergoes an elastic collision. We will consider similar deterministic models in Chapter 8. Instead, we first simulate a probabilistic model based on a *random process*.

The basic assumption of the probabilistic model that we will consider is that the motion of the particles is random. For simplicity, we assume that the particles do not interact with one another so that the probability per unit time that a particle goes through the hole in the partition is the

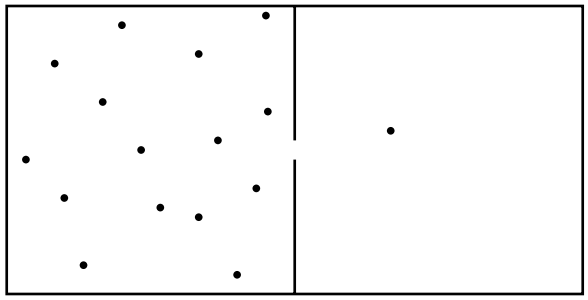

Figure 7.1: A box is divided into two equal halves by a partition. After a small hole is opened in the partition, one particle can pass through the hole per unit time.

same for all particles regardless of the number of particles in either half. We also assume that the size of the hole is such that only one particle can pass through it in one unit of time.

The model consists of N noninteracting particles, all of which are initially on the left-hand side. One way to move a particle is to choose a particle at random and move it to the other side. We could use arrays to specify the position of each particle and simulate the random process by generating at random an integer `i` between 0 and $N-1$ and changing the arrays appropriately. The tool we need to simulate this random process is a random number generator.

It might seem strange that we can use a deterministic computer to generate sequences of random numbers. In Section 7.9 we discuss some of the methods for computing a set of numbers which appear statistically random, but are in fact generated by a deterministic algorithm. These algorithms are sometimes called pseudorandom number generators to distinguish their output from intrinsically random physical processes such as the time between clicks in a Geiger counter near a radioactive sample.

For the present we will be content to use the random number generator supplied with Java, although the random number generators included with various programming languages vary greatly in quality. In Java the method `Math.random()` produces a random number r that is uniformly distributed in the interval $0 \leq r < 1$. To generate a random integer i between 0 and $N-1$, we write:

```
int i = (int)(N*Math.random());
```

The effect of the `(int)` cast is to eliminate the decimal digits from a floating point number. Thus, `(int)(5.7) = 5`;

The algorithm for simulating the evolution of the model can be summarized by the following steps:

1. Use a random number generator to choose a particle at random.
2. Move this particle to the other side of the box.
3. Give the particle a random position on the new side of the box. This step is for visualization purposes only.

4. Increase the "time" by unity.

Note that the above definition of time is arbitrary. Class `Box` implements this algorithm and class `BoxApp` plots the evolution of the number of particles on the left half of the box.

Listing 7.1: Class `Box` for the simulation of the approach to equilibrium.

```
/*
 * The org.opensourcephysics.sip.ch07 package contains classes for Chapter 7,
 * Random Processes, of the book Simulations in Physics.
 * Copyright (c) 2005 H. Gould, J. Tobochnik, and W. Christian.
 */
package org.opensourcephysics.sip.ch07;
import java.awt.Color;
import java.awt.Graphics;
import org.opensourcephysics.display.Drawable;
import org.opensourcephysics.display.DrawingPanel;

/**
 * Box contains data for articles in a divided box
 *
 * @author Jan Tobochnik
 */

public class Box implements Drawable {
        public double x[],y[];
        public int n,nLeft,steps;


  /**
   * Initializes the box, places particles in random positions on left side.
   */

public void initialize () {
    x = new double[n]; //location of molecules
    y = new double[n];
    nLeft = n;  // start with all molecules on the left
    steps = 0;
    for(int i = 0; i < n; i++) {
      x[i] = 0.5*Math.random(); //just needed for visualization
      y[i] = Math.random();
    }
}

  /**
   * Moves one particle to the other side
   */

public void step() {
  int i = (int)(Math.random()*n);
  if(x[i] < 0.5) {
```

```
        nLeft--;  // move to right
        x[i] = 0.5*(1 + Math.random());
        y[i] = Math.random();
    }
    else {
        nLeft++;  // move to left
        x[i] = 0.5*Math.random();
        y[i] = Math.random();
    }
    steps++;
  }

  /**
   * Draws particles and divided box
   */


public void draw (DrawingPanel myWorld, Graphics g) {
  if(x == null) return;
  int size = 2;
  int xm = myWorld.xToPix(0.5);
  g.setColor(Color.black);
  g.drawLine(xm,myWorld.yToPix(0),xm,myWorld.yToPix(0.45));
  g.drawLine(xm,myWorld.yToPix(0.55),xm,myWorld.yToPix(1.0));
  g.setColor(Color.red);
  for(int i = 0; i < n; i++) {
    int xpix = myWorld.xToPix(x[i]);
    int ypix = myWorld.yToPix(y[i]);
    g. fillOval (xpix,ypix, size , size );
  }
}

}
```

Listing 7.2: Target class for plotting the approach to equilibrium.

```
/*
 * The org.opensourcephysics.sip.ch07 package contains classes for Chapter 7,
 * Random Processes, of the book Simulations in Physics.
 * Copyright (c) 2005 H. Gould, J. Tobochnik, and W. Christian.
 */
package org.opensourcephysics.sip.ch07;
import org.opensourcephysics.controls.AbstractSimulation;
import org.opensourcephysics.controls.SimulationControl;
import org.opensourcephysics.frames.DisplayFrame;
import org.opensourcephysics.frames.PlotFrame;
/**
 * Simulates approach to equilibrium for particles in a divided box
 * @author Jan Tobochnik, revised 12/10/04
 */
```

```
public class BoxApp extends AbstractSimulation {
  Box            box = new Box();
  PlotFrame          plotFrame = new PlotFrame("Time", "Number on Left", "Box data");
  DisplayFrame   displayFrame = new DisplayFrame("Divided Box");
  int pSteps; // number of steps between plots

/**
   * Get's parameters and initializes model
   */

  public void initialize () {
     displayFrame.clearDrawables();
     displayFrame.addDrawable(box);
     box.n = control.getInt("Number of Particles");
     pSteps = control.getInt("Number steps between plots");
     box. initialize ();
     plotFrame.clearData();
     displayFrame.setPreferredMinMax(0,1,0,1);
  }

/**
   * Draws particles after each move, and plots data after pSteps steps
   */

  public void doStep() {
    for(int i = 0; i < pSteps;i++) {
      box.step();
    }
    plotFrame.append(0, box.steps, box.nLeft);
  }

 /**
   * Resets to default values
   */


 public void reset() {
    control.setValue("Number of Particles", 100);
    control.setValue("Number steps between plots", 10);
    plotFrame.clearData();
 }

   /**
   * Starts the Java application.
   * @param args command line parameters
   */

  public static void main (String[] args) {
      SimulationControl.createApp(new BoxApp());
```

```
    }

  }
```

How long does it take for the system to reach equilibrium? How does this time depend on the number of particles? After the system reaches equilibrium, what is the magnitude of the fluctuations? How do the fluctuations depend on the number of particles? Problems 7.2 and 7.3 address such questions.

Exercise 7.1. Some simple tests

a. It is frequently quicker to write a short program to test the properties of a function than to look it up in a manual or online. Write a short program to determine the values of `3/2`, `3.0/2.0`, `(int)(3.0/2.0)`, `2/3`, and `-3/2`.

b. Write a short target class to determine the nature of `Math.round(arg)`, `Math.ceil(arg)`, and `Math.rint(arg)`.

c. Write a program to test whether the same sequence of random numbers appears each time the program is run if we use the method `Math.random()` to generate the sequence. In Section 7.9 we will learn about ways of generating random numbers and how we can set the seed in Java so that we can generate the same random number sequence if we wish. The ability to do the latter is essential for testing our programs.

Problem 7.2. Approach to equilibrium

a. Run the program and describe the evolution of n, the number of particles on the left side of the box. Choose the total number of particles N to be $N = 8$, 16, 64, 400, 800, and 3600. What is your qualitative criterion for equilibrium? Does n, the number of particles on the left-hand side, change when the system is in equilibrium?

b. The algorithm we have used is needlessly cumbersome, because our only interest is the number of particles on each side, not their position. We used the positions only for visualization purposes. Because each particle has the same chance to go through the hole, the probability per unit time that a particle moves from left to right equals the number of particles on the left divided by the total number of particles, that is, $p = n/N$. Modify the program so that the following algorithm is implemented.

   (i) Generate a random number r from a uniformly distributed set of random numbers in the interval $0 \leq r < 1$.

   (ii) If $r \leq p = n/N$, move a particle from left to right, that is $n \to n-1$; otherwise, $n \to n+1$.

c. For sufficiently large N, does the time dependence of n appear to be deterministic? Based on the shape of your plots of $n(t)$, what is the qualitative behavior of $n(t)$ before equilibrium is reached?

d. Estimate the time for the system to reach equilibrium from the plots. How does this time depend on N?

Problem 7.3. Equilibrium fluctuations

a. As a rough measure of the equilibrium fluctuations, visually estimate the deviation of $n(t)$ from $N/2$ for $N = 16$, 64, 400, 800, and 3600? Choose a time interval that is bigger than the time needed to reach equilibrium. How do your results for the deviation depend on N?

b. A better measure of the equilibrium fluctuations is the variance σ_n^2, which is defined as

$$\sigma_n^2 = \overline{(n-\overline{n})^2} = \overline{n^2 - n\overline{n} - \overline{n}n + \overline{n})^2} = \overline{n^2} - 2\overline{n}\,\overline{n} + \overline{n}^2 = \overline{n^2} - \overline{n}^2. \tag{7.1}$$

The bar denotes an average taken after the system has reached equilibrium. The relative magnitude of the fluctuations is $\sigma_n/\overline{n}$. Modify your program so that averages are taken after equilibrium has been reached. Run for a time that is long enough to obtain meaningful results. Compute the variance σ_n^2 for the same values of N considered in part (a). How do the relative fluctuations, $\sigma_n/\overline{n}$, depend on N? (You might find it helpful to see how averages are computed in Listings 7.3 and 7.4.)

From Problem 7.2 we see that $n(t)$ decreases in time from its initial value to its equilibrium value in an almost deterministic manner if $N >> 1$. It is instructive to derive the time dependence of $n(t)$ to show explicitly how chance can generate deterministic behavior. If there are $n(t)$ particles on the left side after t moves, then the change in $\overline{n}(t)$ in the time interval Δt is given by

$$\Delta\overline{n} = \Big[\frac{-\overline{n}(t)}{N} + \frac{N-\overline{n}(t)}{N}\Big]\Delta t. \tag{7.2}$$

(We defined the time so that the time interval $\Delta t = 1$ in our simulations.) What is the meaning of the two terms in (7.2)? If we treat $\overline{n}$ and t as continuous variables and take the limit $\Delta t \to 0$, we have

$$\frac{\Delta\overline{n}}{\Delta t} \to \frac{d\overline{n}}{dt} = 1 - \frac{2\overline{n}(t)}{N}. \tag{7.3}$$

The solution of the differential equation (7.3) is

$$\overline{n}(t) = \frac{N}{2}[1 + e^{-2t/N}], \tag{7.4}$$

where we have used the initial condition $\overline{n}(t=0) = N$. Note that $\overline{n}(t)$ decays exponentially to its equilibrium value $N/2$. How does this form (7.4) compare to your simulation results for various values of N? We can define a *relaxation time* τ as the time it takes the difference $[\overline{n}(t) - N/2]$ to decrease to $1/e$ of its initial value. How does τ depend on N? Does this prediction for τ agree with your results from Problem 7.2?

***Problem 7.4.** The effect of a simple modification

Modify your program so that each side of the box is chosen with equal probability. One particle is then moved from the side chosen to the other side. If the side chosen does not have a particle, then no particle is moved during this time interval. Do you expect that the system behaves in the same way as before? Do the simulation starting with all the particles on the left side of the box and choose $N = 800$. To simplify the code, do not keep track of the particle's positions. Compare the behavior of $n(t)$ with the behavior of $n(t)$ found in Problem 7.3. How do the values of $\overline{n}$ and σ_n^2 compare? Is this variation of the model realistic?

The probabilistic method discussed on page 202 for simulating the approach to equilibrium is an example of a *Monte Carlo* algorithm, that is, the random sampling of the most probable outcomes. An alternative method is to use *exact enumeration* and to determine all the possibilities at each time interval. For example, suppose that $N = 8$ and $n(t = 0) = 8$. At $t = 1$, the only possibility is $n = 7$ and $n' = 1$. Hence, $P(n = 7, t = 1) = 1$ and all other probabilities are zero. At $t = 2$, one of the seven particles on the left can move to the right, or the one particle on the right can move to the left. Because the first possibility can occur in seven different ways, we have the nonzero probabilities, $P(n = 6, t = 2) = 7/8$ and $P(n = 8, t = 2) = 1/8$. Hence at $t = 2$, the average number of particles on the left side of the box is

$$\langle n(t = 2)\rangle = 6P(6, 2) + 8P(8, 2) = \frac{1}{8}[6 \times 7 + 8 \times 1] = 6.25.$$

We have denoted the average by the brackets $\langle \ldots \rangle$ to distinguish it from the time average that was computed in Problem 7.3. Is this exact result consistent with what you found in Problem 7.2? In this example N is small, and we can continue the enumeration of all the possibilities indefinitely. However for larger N, the number of possibilities becomes very large after a few time intervals, and we are forced to use Monte Carlo methods to sample the most probable outcomes.

So far you probably have run each simulation only once. If we are interested only in estimating various averages after the system has reached equilibrium, we can obtain good averages by running for a sufficiently long time. However, each time we run the program with a different sequence of random numbers, we expect that $n(t)$ will be somewhat different. So if we are interested in an accurate estimate of the approach to equilibrium, we would need to run the program many times with a different random number seed and average our results.

7.2 Random Walks

In Section 7.1 we considered the random motion of many particles in a box, but we did not care about their trajectories – all we needed to know was the number of particles on each side. Suppose that we want to characterize the motion of a dust particle in the atmosphere. We know that as a given dust particle collides with the other molecules, it changes its direction frequently, and its motion appears to be random. A simple model for the trajectory of a dust particle in the atmosphere is based on the assumption that the particle moves in any direction with equal probability. Such a model is an example of a *random walk*.

The original statement of a random walk was formulated in the context of a drunken sailor. If a drunkard begins at a lamp post and takes N steps of equal length in random directions, how far will the drunkard be from the lamp post? We will find that the mean square displacement of a random walker, for example, a dust particle or a drunkard, grows linearly with time. This result and its relation to diffusion leads to many applications that might seem to be unrelated to the original drunken sailor problem.

We first consider an idealized example of a random walker that can move only along a line. Suppose that the walker begins at $x = 0$ and that each step is of equal length a. At each time interval the walker has a probability p of a step to the right and a probability $q = 1 - p$ of a step to the left. The direction of each step is independent of the preceding one. (The sailor is very

drunk.) After N steps the displacement x of the walker from the origin is given by

$$x_N = \sum_{i=1}^{N} s_i, \tag{7.5}$$

where $s_i = \pm a$. The displacement squared is

$$x_N^2 = \big[\sum_{i=1}^{N} s_i\big]^2. \tag{7.6}$$

We can generate one walk of N steps by flipping a coin N times and increasing x by a each time the coin is heads and decreasing x by a each time the coin is tails. In this case $p = q = \frac{1}{2}$.

In general, we would expect that if we average over a sufficient number of N step walks, then the average of x_N, denoted by $\langle x_N \rangle$, would be $(p-q)Na$. We can derive this result analytically, by writing $\langle x_N \rangle = \sum_{i=1}^{N} \langle s_i \rangle = N\langle s \rangle$, where we have assumed that each step is the same on the average. We have $\langle s \rangle = p(a) + q(-a) = (p-q)a$, and the result follows. For simplicity, we will frequently omit the subscript N and write $\langle x \rangle$.

To determine $\langle x_N^2 \rangle$ analytically, we write (7.6) as the sum of two terms:

$$x_N^2 = \sum_{i=1}^{N} s_i \sum_{j=1}^{N} s_j = \sum_{i=1}^{N} s_i^2 + \sum_{i \neq j=1}^{N} s_i s_j. \tag{7.7}$$

The first sum on the right-hand side of (7.7) includes terms for which $i = j$; the second sum is over i and j such that $i \neq j$. The product $s_i s_j$ for $i \neq j$ equals $+a^2$ and $-a^2$ with equal probability, and hence the average of the second term in (7.7) is zero. (Remember that the average of a sum is the sum of the averages.) Because $s_i^2 = a^2$ independently of the sign of s_i, the first term in (7.7) equals $a^2 N$ for all walks, and hence equals $a^2 N$ on the average. We conclude that

$$\langle x_N^2 \rangle = a^2 N. \qquad (p = q = 1/2) \tag{7.8}$$

If the time interval for a step is Δt rather than unity, we should replace N in (7.8) by $N\Delta t$.

For $p \neq \frac{1}{2}$, it is convenient to consider the variance σ_x^2 defined as

$$\sigma_x^2 = \langle \big[x_N - \langle x_N \rangle\big]^2 \rangle = \langle x_N^2 \rangle - \langle x_N \rangle^2. \tag{7.9}$$

It is straightforward to show that for any value of p, the N dependence of σ_x^2 is given by

$$\sigma_x^2 = 4pqa^2 N. \tag{7.10}$$

W can gain more insight into the nature of random walks by doing a Monte Carlo simulation, that is, by using a computer to "flip coins." The implementation of the random walk algorithm is simple, for example,

```
if (p < Math.random()) {
   x++;
}
else {
   x--;
}
```

Clearly we have to sample many N step walks because in general, each walk will give a different outcome. We need to do a Monte Carlo simulation many times and average over the results to obtain meaningful averages. Each N-step walk is called a *trial*. How do we know how many trials to use? The simple answer is to average over more and more trials until the average results don't change within the desired accuracy. The more sophisticated answer is to do an error analysis similar to what we do in measurements in the laboratory. Such an analysis is discussed in Section 11.5.

The more difficult parts of a program to simulate random walks are associated with bookkeeping. The walker takes a total of N steps in each trial and the net displacement x is computed. Our convention will be to use a variable name ending in `Accumulator` (or `Accum`) to denote a variable that accumulates the value of some variable. Below we provide classes that can be used to simulate random walks.

Listing 7.3: Listing of `Walker` class.

```
/*
 * The org.opensourcephysics.sip.ch11 package contains classes for chapter 11,
 * Random Processes, of the book Simulations in Physics.
 * Copyright (c) 2002 H. Gould, J. Tobochnik, and W. Christian.
 */
package org.opensourcephysics.sip.ch07;


/**
 * Random Walk simulation in 1-D
 *
 * @author Jan Tobochnik
 */

public class Walker {

        int x[], x2[];     // accumulated data on location of walkers, index is time
        int tMax;          // maximum time to record data
        double p;          // probability of walking to the right
        int position;      // walker position


  /**
   * Initializes walker array
   */

public void initialize () {
    x = new int[tMax+1];
    x2 = new int[tMax+1];
}

  /**
   * Random walk for one walker
   */
```

```
public void step() {
    position = 0;
    for(int t = 0; t < tMax; t++) {
        if(Math.random() < p)
          position++;
        else
          position--;
        x[t+1] += position;
        x2[t+1] += position*position;
    }
}

}
```

Listing 7.4: Target class for random walk simulation.

```
package org.opensourcephysics.sip.ch07;
import org.opensourcephysics.controls.*;
import org.opensourcephysics.frames.*;

/**
 * Simulates random walkers in one dimension
 * @author Jan Tobochnik revised 12/10/04
 */
public class WalkerApp extends AbstractSimulation {
    Walker walker = new Walker();
    PlotFrame plotFrame = new PlotFrame("time", "<x>,<x^2>", "Averages");
    HistogramFrame distribution = new HistogramFrame("x", "H(x)", "Histogram");
    int trials ;    // number of trials

    /**
     *   Set column names for data table
     */
    public WalkerApp() {
        plotFrame.setXYColumnNames(0, "t", "<x>");
        plotFrame.setXYColumnNames(1, "t", "<x^2>");
    }

    /**
     *   Get's parameters and initializes model
     */
    public void initialize () {
        walker.p = control.getDouble("Probability of walking right");
        walker.tMax = control.getInt("Maximum time to collect data");
        walker. initialize ();
        trials = 0;
    }

    /**
     *   Does one walker at a time
```

```
     */
    public void doStep() {
        trials ++;
        walker.step();
        distribution.append(walker.position);
        distribution.setMessage("trials = "+trials);
    }

    /**
     * Plots data when user stops the simulation.
     */
    public void stopRunning() {
        plotFrame.clearData();
        for(int t = 0; t<=walker.tMax; t++) {
            plotFrame.append(0, 1.0*t, walker.x[t]*1.0/ trials );
            plotFrame.append(1, 1.0*t, walker.x2[t]*1.0/ trials -Math.pow(walker.x[t]/trials, 2));
        }
        plotFrame.repaint();
    }

    /**
     * Resets to default values
     */
    public void reset() {
        control.setValue("Probability of walking right" , 0.5);
        control.setValue("Maximum time to collect data", 100);
    }

    /**
     * Starts the Java application.
     * @param args command line parameters
     */
    public static void main(String[] args) {
        SimulationControl.createApp(new WalkerApp());
    }
}
```

Problem 7.5. Random walks in one dimension

a. In class `Walker` the steps are of unit length so that $a = 1$. Use `Walker` and `WalkerApp` to estimate the number of trials needed to obtain σ_x^2 for $N = 20$ and $p = 1/2$ with an accuracy of approximately 5%. Compare your result for σ_x^2 to the exact answer in (7.10). Approximately how many trials do you need to obtain the same relative accuracy for $N = 100$?

b. Is $\langle x \rangle$ exactly zero in your simulations? Explain the difference between the analytical result and the results of your simulations.

c. How do your results for $\langle x \rangle$ and σ_x^2 change for $p \neq q$? Choose $p = 0.7$ and determine the N dependence of $\langle x \rangle$ and σ_x^2.

d.* Determine σ_x^2 for $N = 1$ to $N = 5$ by enumerating all the possible walks. For simplicity, choose $p = \frac{1}{2}$ so that $\langle x \rangle = 0$. For $N = 1$, there are two possible walks: one step to the right and one step to the left. In both cases $x^2 = 1$ and hence $\langle {x_1}^2 \rangle = 1$. For $N = 2$ there are four possible walks with the same probability: (i) two steps to the right, (ii) two steps to the left, (iii) first step to the right and second step to the left, and (iv) first step to the left and second step to the right. The value of ${x_2}^2$ for these walks is 4, 4, 0, and 0 respectively, and hence $\langle {x_2}^2 \rangle = (4 + 4 + 0 + 0)/4 = 2$. Write a program that enumerates all the possible walks of a given number of steps and compute the various averages of interest exactly.

The class `WalkerApp` displays the distribution of values of the displacement x after N steps. One way of determining the number of times that the variable x has a certain value would be to define a one-dimensional array, `probability`, and let

```
probability[x] += 1;
```

In this case because x takes only integer values, the array index of `probability` is the same as x itself. However, the above statement does not work in Java because x can be negative as well as positive. What we need is a way of mapping the value x to a bin or index number. The `HistogramFrame` class, which is part of the Open Source Physics display package, does this mapping automatically using the Java `Hashtable` class. In many data structures data is accessed by an index that indicates the location of the data in the data structure. However, hashtable data is accessed by a *key*, which in our case would be the value of x. A hashing function converts the key to an index. The `append` method of the `HistogramFrame` class takes a value, finds the index using a hashing function, and then increments the data associated with that key. The `HistogramFrame` class also draws itself.

The `HistogramFrame` class is very useful for taking a quick look at the distribution of values in a data set. You don't need to worry about how to group the data into bins or the range of values of the data. The default bin width is unity, but the bin width can be set using the `setBinWidth` method. Frequently, we wish to use the histogram data to compute other quantities such as a probability density. You could collect the data using the Data Table menu item in `HistogramFrame` and copy the data to a plotting program. Another option is to include additional code in your program to analyze the data. You can use the following statements, which assume that a `HistogramFrame` object called `histogram` has been created and data entered into it.

```
// creates array entries of data from histogram
java.util.Map.Entry[] entries = histogram.entries();
for(int i = 0, length = entries.length; i < length; i++) {
   Integer binNumber = (Integer) entries[i].getKey();    // gets bin number
   Double occurences = (Double) entries[i].getValue(); // gets number of occurences for bin number i
   double value = histogram.getLeftMostBinPosition(binNumber.intValue()); // gets value of left edge of bin
   value += 0.5*histogram.getBinWidth(); // sets value to middle of bin
   double number = occurences.doubleValue(); // convert from Double class to double data type
   // use value and number in your analysis
}
```

Problem 7.6. Nature of the probability distribution

a. Compute $P_N(x)$, the probability that the displacement of the walker from the origin is x after N steps. What is the difference between the histogram, that is, the number of occurrences, and the probability? Consider $N = 10$ and $N = 40$ and at least 1000 trials. Does the qualitative form of $P_N(x)$ change as the number of trials increases? What is the approximate width of $P_N(x)$ and the value of $P_N(x)$ at its maximum for each value of N?

b. Is $P_N(x)$ a continuous function of x? Can you fit the envelope of $P_N(x)$ to a continuous function such as

$$\frac{1}{\sqrt{2\pi\sigma_x^2}} e^{-(x-\langle x\rangle)^2/2\sigma_x^2)}. \tag{7.11}$$

The form (7.11) is the standard form of the Gaussian distribution. Compare your computed values for $P_N(x)$ to the form (7.11) using $\langle x\rangle$ and σ_x^2 as input.

Problem 7.7. More random walks in one dimension

a. Suppose that the probability of moving to the right is $p = 0.7$. Compute $\langle x\rangle$ and σ_x^2 for $N = 4$, 8, 16, and 32. What is the interpretation of $\langle x\rangle$ in this case? What is the qualitative dependence of σ^2 on N?

b. Use the exact result and estimate the number of trials needed to obtain σ_x^2 to 1% accuracy for $N = 8$ and $N = 32$.

c. An interesting property of random walks is the mean number $\langle D_N\rangle$ of *distinct* lattice sites visited during the course of an N step walk. Do a Monte Carlo simulation of $\langle D_N\rangle$ and determine its N dependence.

We can equally well consider either a large number of successive walks as in Problem 7.7 or a large number of similar (noninteracting) walkers moving at the same time. In Problem 7.8 we consider the motion of many random walkers moving independently of one another on a two-dimensional lattice.

Problem 7.8. A random walk in two dimensions

a. Consider a collection of walkers initially at the origin of a square lattice. At each unit of time, each of the walkers moves at random with equal probability in one of the four possible directions. Modify `WalkerApp` and create a drawable class, `Walker2D`, which contains the positions of M walkers moving in two dimensions and draws their location. Unlike `WalkerApp`, this new class should not specify the maximum number of steps. Instead the number of walkers should be specified.

b. Run your application with the number of walkers $M \geq 200$ and allow the walkers to take at least 500 steps. If each walker represents a bee, describe the qualitative nature of the shape of the swarm of bees. Describe the qualitative nature of the surface of the swarm as a function of the number of steps, N. Is the surface jagged or smooth?

c. Compute the quantities $\langle x_N \rangle$, $\langle y_N \rangle$, σ_x^2, and σ_y^2 as a function of N. The average is over the M walkers. Also compute the mean square displacement $\langle R_N^2 \rangle$ given by

$$\langle R_N^2 \rangle = \langle x_N^2 \rangle - \langle x_N \rangle^2 + \langle y_N^2 \rangle - \langle y_N \rangle^2 = \sigma_x^2 + \sigma_y^2. \tag{7.12}$$

What is the dependence of each quantity on N? (As before, we will frequently write $\langle R^2 \rangle$ instead of $\langle R_N^2 \rangle$.)

d. Estimate $\langle R^2 \rangle$ for $N = 8$, 16, 32, and 64 using a large number of walkers for each value of N. Assume that $\langle R^2 \rangle$ has the asymptotic N dependence:

$$\langle R^2 \rangle \sim N^{2\nu}, \qquad (N >> 1) \tag{7.13}$$

and estimate the exponent ν from a log-log plot of $\langle R^2 \rangle$ versus N. We will see in Chapter 13 that the exponent ν is related to the fractal dimension and that its value is related to how a random walk fills space. If $\nu \approx \frac{1}{2}$, estimate the magnitude of the self-diffusion coefficient D given by

$$\langle R^2 \rangle \to 2dDN, \qquad (N >> 1) \tag{7.14}$$

where $d = 2$ is the dimension of space.

e. Do a Monte Carlo simulation of $\langle R^2 \rangle$ on a triangular lattice (see Figure **??**) and estimate ν. Can you conclude that the exponent ν is independent of the symmetry of the lattice? Does D depend on the symmetry of the lattice? If so, give a qualitative explanation for this dependence.

f.* Enumerate all the random walks on a square lattice for $N = 4$ and obtain exact results for $\langle x \rangle$, $\langle y \rangle$, and $\langle R^2 \rangle$. Assume that all four directions are equally probable. Verify your program by comparing the Monte Carlo and exact enumeration results.

Problem 7.9. The fall of a rain drop

Consider a random walk that starts at a site a distance $y = h$ above a horizontal line (see Figure 7.2). If the probability of a step down is greater than the probability of a step up, we expect that the walker will eventually reach a site on the horizontal line. This walk is a simple model of the fall of a rain drop in the presence of a random swirling breeze. Do a Monte Carlo simulation to determine the mean time τ for the walker to reach any site on the line $y = 0$ and find the functional dependence of τ on h. Is it possible to define a velocity in the vertical direction? Because the walker does not always move vertically, it suffers a net displacement x in the horizontal direction. How does σ_x^2 depend on h and τ? Reasonable values for the step probabilities are 0.1, 0.6, 0.15, 0.15, corresponding to up, down, right, and left, respectively.

7.3 Modified Random Walks

So far we have considered random walks on one- and two-dimensional lattices where the walker has no "memory" of the previous step. What happens if the walkers remember the nature of their previous steps? What happens if there are multiple random walkers, with the condition that no

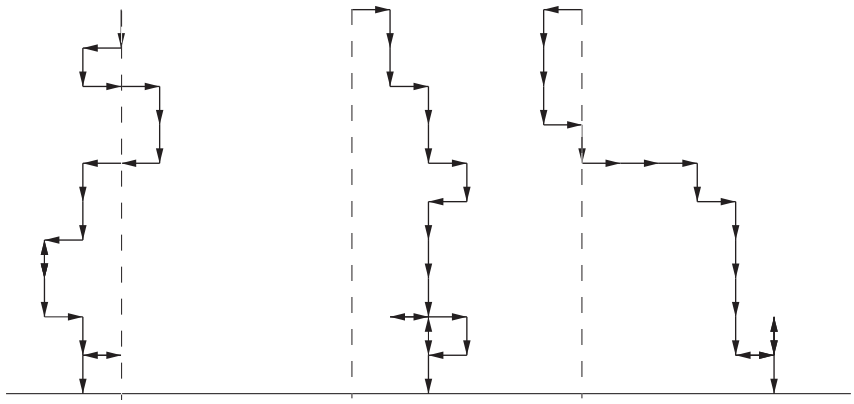

Figure 7.2: Examples of the random path of a raindrop to the ground. The step probabilities are given in Problem 7.9. The walker starts at $x = 0$, $y = h$.

double occupancy is allowed? We explore these and other variations of the simple random walk in this section. All these variations have applications to physical systems, but the applications are more difficult to understand than the models themselves.

Problem 7.10. A persistent random walk

a. In a persistent random walk, the *transition* or jump probability depends on the previous step. Consider a walk on a one-dimensional lattice, and suppose that step $N-1$ has been made. Then step N is made in the same direction with probability α; a step in the opposite direction occurs with probability $1-\alpha$. Write a program to do a Monte Carlo simulation of the persistent random walk in one dimension. Compute $\langle x \rangle$, σ_x^2, and $P_N(x)$. Note that it is necessary to specify both the initial position and an initial direction of the walker. What is the $\alpha = \frac{1}{2}$ limit of the persistent random walk?

b. Consider the cases $\alpha = 0.25$ and $\alpha = 0.75$ and determine σ_x^2 for $N = 8$, 64, 256, and 512. Assume that $\sigma_x^2 \sim N^{2\nu}$ for large N, and estimate the value of ν from a log-log plot of σ_x^2 versus N for large N. Does ν depend on α? If $\nu \approx \frac{1}{2}$, determine the self-diffusion coefficient D for $\alpha = 0.25$ and 0.75 (see Problem 7.8). Give a physical argument why $D(\alpha \neq 0.5)$ is greater (smaller) than $D(\alpha = 0.5)$.

c. You might have expected that the persistent random walk yields a nonzero value for $\langle x_N \rangle$. Verify that $\langle x_N \rangle = 0$, and explain why this result is exact. How does the persistent random walk differ from the biased random walk for which $p \neq q$?

d. A persistent random walk can be considered as an example of a *multistate* walk in which the state of the walk is defined by the last transition. In the above example, the walker is in one of two states; at each step the probabilities of remaining in the same state or switching states are α and $1-\alpha$, respectively. One of the earliest applications of a two state random walk was to the study of diffusion in a chromatographic column. Suppose that a molecule in a

chromatographic column can be either in a mobile phase (constant velocity v) or in a trapped phase (zero velocity). Instead of each step changing the position by ± 1, the position at each step changes by $+v$ or 0. A quantity of experimental interest is the probability $P_N(x)$ that a molecule has traveled a distance x in N steps. Choose $v = 1$ and $\alpha = 0.75$ and compute the qualitative behavior of $P_N(x)$.

e.* A molecule is said to *diffuse* if its mean square displacement is proportional to the number of steps (or time). Explain why the molecule cannot diffuse in either state, but that it is still possible to define an effective diffusion coefficient for the molecule.

The fall of a raindrop considered in Problem 7.9 is an example of a *restricted* random walk, that is, a walk in the presence of a boundary. In the following problem, we discuss in a more general context the effects of various types of restrictions or boundaries on random walks. Other examples of a restricted random walk are given in Problems 7.23b and 7.17.

Problem 7.11. Restricted random walks

a. Consider a one-dimensional lattice with trap sites at $x = 0$ and $x = L\,(L > 0)$. A walker begins at site x_0 $(0 < x_0 < L)$ and takes unit steps to the left and right with equal probability. When the walker arrives at a trap, it vanishes. Do a Monte Carlo simulation and verify that the mean number of steps τ for the particle to be trapped (*the first passage time*) is given by

$$\tau = (2D)^{-1} x_0 (L - x_0). \tag{7.15}$$

D is the self-diffusion coefficient in the absence of the traps, and the average is over all possible walks.

b. Random walk models in the presence of traps have had an important role in condensed matter science. For example, consider the following idealized model of energy transport in solids. The solid is represented as a lattice with two types of sites: hosts and traps. An incident photon is absorbed at a host site and excites the host molecule or atom. The excitation energy or *exciton* is transferred at random to one of the host's nearest neighbors and the original excited molecule returns to its ground state. In this way the exciton wanders through the lattice until it reaches a trap site. The exciton is then trapped and a chemical reaction occurs. A simple version of this energy transport model is given by a one-dimensional lattice with traps placed on a periodic sublattice. Because the traps are placed at regular intervals, we can replace the random walk on an infinite lattice by a random walk on a circular ring. Consider a lattice of N host or nontrapping sites and one trap site. If a walker has an equal probability of starting from any host site and an equal probability of a step to each nearest neighbor site, what is the N dependence of the mean survival time τ (the mean number of steps taken before a trap site is reached)? Use the results of part (a) rather than doing another simulation.

c. Consider a one-dimensional lattice with reflecting sites at $x = -L$ and $x = L$. For example, if a walker reaches the reflecting site at $x = L$, it is reflected at the next step to $x = L - 1$. At $t = 0$, the walker starts at $x = 0$ and steps with equal probability to nearest neighbor sites. Write a Monte Carlo program to determine $P_N(x)$, the probability that the walker is at site x after N steps. Compare the form of $P_N(x)$ with and without the presence of the reflecting "walls." Can you distinguish the two probability distributions if N is the order of L? At what value of N can you first distinguish the two distributions?

Problem 7.12. Synchronized random walks

a. Randomly place two walkers on a one-dimensional lattice of L sites. At each time step randomly choose whether the walkers should move to the left or to the right. Both walkers move in the same direction. If a walker cannot move in the chosen direction because it is at a boundary, then this walker stays still for this time step. A trial ends when both walkers are at the same site. This model is relevant to a method of doing crytography using neural networks (see Rutter et al.). Write a program to determine the mean time and the variance of the time for two walkers to reach the same site.

b. Start with $L = 1$, and continuing increasing L by unity until the user stops the simulation. Use at least 1000 trials for each value of L and determine the dependence of the mean time and the variance of the time on L.

c. Change your program so that you use biased random walkers for which $p \neq q$. How does this change affect your results?

Problem 7.13. Random walk on a continuum

One of the first continuum models of a random walk was proposed by Rayleigh in 1919. The model is known as the freely jointed chain in polymer physics. In this model the length a of each step is a random variable with probability density $p(a)$, and the direction of each step is uniformly random. For simplicity, we first consider a walker in two dimensions so that each step has unit length at a random angle. Write a Monte Carlo program to compute $P_N(r)\Delta r$, the probability that the displacement of the walker is in the range r to $r+\Delta r$ after N steps, where r is the distance from the origin. Verify that the probability density $P_N(r)$ can be approximated by a Gaussian for sufficiently large N. Is a Gaussian a good approximation for small N? Is it necessary to do a Monte Carlo simulation to confirm that $\langle R_N^2 \rangle \sim N$, or can you give a simple argument for this dependence based on the form of $P_N(r)$?

Problem 7.14. Random walks with steps of variable length

a. Consider a random walk in one dimension with jumps of all lengths allowed. The probability density that the length of a single step is a is $p(a)$. If the form of $p(a)$ is given by $p(a) = e^{-a}$ for $a > 0$, what is the form of $P_N(x)$? As we will learn in Section 11.6, the code needed to generate step lengths according to this probability density is given by

```
a = −Math.log(1 − Math.random());
```

Consider a walk of N steps and determine the net displacement x. Generate many such walks and determine $P_N(x)$. Plot $P_N(x)$ versus x and confirm that the form of $P_N(x)$ is consistent with a Gaussian distribution.

b. Assume that the probability density $p(a)$ is given by $p(a) = C/a^2$ for $a \geq 1$. Determine the normalization constant C using the condition $C \int_1^\infty a^{-2}\, da = 1$. Does the second moment of $p(a)$ exist? In this case, we will learn that the statement

```
a = 1.0/(1.0 − Math.random());
```

generates values of a according to this form of $p(a)$. Do a Monte Carlo simulation as in part (a) and verify that the form of $P_N(x)$ is not a Gaussian, but is given by

$$P_N(x) \sim \frac{bN}{x^2 + b^2N^2}. \tag{7.16}$$

What is the magnitude of the parameter b? Does the variance σ_x^2 of $P_N(x)$ exist? This type of random walk for which $p(a)$ decreases as a power law, $a^{-1-\alpha}$, is known as a Levy flight for $\alpha \leq 2$.

Problem 7.15. The central limit theorem

Consider a continuous random variable x with probability density $f(x)$. That is, $f(x)\Delta x$ is the probability that x has a value between x and $x + \Delta x$. The mth *moment* of $f(x)$ is defined as

$$\langle x^m \rangle = \int x^m f(x)\, dx. \tag{7.17}$$

The mean value $\langle x \rangle$ is given by (7.17) with $m = 1$. The variance ${\sigma_x}^2$ of $f(x)$ is defined as

$${\sigma_x}^2 = \langle x^2 \rangle - \langle x \rangle^2. \tag{7.18}$$

Consider the sum y_N corresponding to the average of N values of x:

$$y_N = \frac{1}{N}(x_1 + x_2 + \ldots + x_N). \tag{7.19}$$

We adopt the notation $y = y_N$. Suppose that we make many measurements of y. We know that the values of y will not be identical, but will be distributed according to a probability density $p(y)$, where $p(y)\Delta y$ is the probability that the measured value of y is in the range y to $y + \Delta y$. The main quantities of interest are the mean $\langle y \rangle$, the variance ${\sigma_y}^2 = \langle y^2 \rangle - \langle y \rangle^2$, and $p(y)$.

a. Suppose that $f(x)$ is uniform in the interval $[-1, 1]$. Calculate $\langle x \rangle$ and σ_x analytically. Use a Monte Carlo method to make a sufficient number of measurements of y to determine $p(y), \langle y \rangle$, and σ_y with reasonable accuracy. For example, choose $N = 1000$ and make 10^4 measurements of y. Verify that σ_y is approximately equal to $\sigma_x/\sqrt{N}$. Use the `HistogramFrame` class to plot $p(y)$ versus y and discuss its qualitative form. Does the form of $p(y)$ change if the number of measurements of y is increased? Does the form of $p(y)$ change significantly if N is increased?

b. To test the generality of the results of part (a), consider the exponential probability density

$$f(x) = \begin{cases} e^{-x}, & x \geq 0 \\ 0, & x < 0. \end{cases} \tag{7.20}$$

Calculate $\langle x \rangle$ and σ_x analytically. Modify your Monte Carlo program (see Problem 7.14a) and estimate $\langle y \rangle$, σ_y, and $p(y)$. Is σ_y related to σ_x and N as in part (a)? Plot $p(y)$, and discuss its qualitative form and its dependence on N and on the number of measurements of y.

Problem 7.15 illustrates the *central limit theorem* which states that the probability distribution of a sum of random variables, the random variable y, is a Gaussian centered at $\langle y \rangle$ with a standard deviation $1/\sqrt{n}$ times the standard deviation of $f(x)$. The requirements are that $f(x)$ has finite first and second moments, that the measurements of y are statistically independent, and that N is large. What is the relation of the central limit theorem to the calculations of the probability distribution in the random walk models that we already have considered?

Problem 7.16. Generation of the Gaussian distribution

Consider the sum

$$y = \sum_{i=1}^{12} r_i, \tag{7.21}$$

where r_i is a uniform random number in the unit interval. Make many measurements of y and show that the probability distribution of y approximates the Gaussian distribution with mean value 6 and variance 1. What is the relation of this result to the central limit theorem? Discuss how to use this result to generate a Gaussian distribution with arbitrary mean and variance. This way of generating a Gaussian distribution is particularly useful when a "quick and dirty" approximation is appropriate. The Box-Muller method of generating a set of random numbers distributed according to a Gaussian distribution is discussed in Section 11.6.

Many of the problems we have considered have revealed the slow convergence of Monte Carlo simulations and the difficulty of obtaining quantitative results for asymptotic quantities. We conclude this section with a cautionary note and consider a "simple" problem for which straightforward Monte Carlo methods give misleading asymptotic results.

***Problem 7.17.** Random walk on lattices containing random traps

a. We have considered the mean survival time of a one-dimensional random walker in the presence of a periodic distribution of traps (see Problem 7.11). Now suppose that the trap sites are distributed at *random* on a one-dimensional lattice with density $\rho = N/L$. For example, if $\rho = 0.01$, the probability that site is a trap site is 1%. If a walker is placed at random at any nontrapping site, determine its mean survival time τ, that is, the mean number of steps before a trap site is reached. Assume that the walker has an equal probability of moving to a nearest neighbor site at each step and use periodic boundary conditions, that is, the lattice sites are located on a ring.

   This problem is more difficult than it might first appear, and there are a number of pitfalls. The major complication is that it is necessary to perform *three* averages: the distribution of traps, the origin of the walker, and the different walks for a given trap distribution and origin. Choose reasonable values for the number of trials associated with each average and do a Monte Carlo simulation to estimate the mean survival time τ. If τ exhibits a power law dependence on ρ, for example, $\tau \approx \tau_0\, \rho^{-z}$, estimate the exponent z.

b. A seemingly straightforward extension of part (a) is to estimate the survival probability S_N after N steps. Choose $\rho = 0.5$ and do a Monte Carlo simulation of S_N for N as large as possible. (Published results are for a maximum $N = 3 \times 10^4$ on lattices large enough that a

walker doesn't reach the boundary, and about 54 000 trials.) Assume that the asymptotic form of S_N for large N is given by

$$S_N \sim e^{-bN^\alpha}, \tag{7.22}$$

where b is a constant that depends on ρ. Are your results consistent with this form? Is it possible to make a meaningful estimate of the exponent α?

c. The object of part (b) is to convince you that it is not possible to use Monte Carlo methods directly to obtain the correct asymptotic behavior of S_N. The difficulty is that we are trying to estimate S_N in the asymptotic region where S_N is very small, and the small number of trials in this region prevent us from obtaining meaningful results. It has been proved using analytical methods, that the asymptotic N dependence of S_N has the form (7.22) with $\alpha = 1/3$. Are your Monte Carlo results consistent with this value of α?

d. A method that reduces the number of required averages is to determine exactly, for a given distribution of trap sites, the probability that the walker is at site i after N steps. The method is illustrated in Figure 7.3. The first line represents a given configuration of traps distributed randomly on a one-dimensional lattice. One walker is placed at each non-trap site; trap sites are assigned the value 0. Because each walker moves with probability $\frac{1}{2}$ to each neighbor, the number of walkers $W_i(N+1)$ on site i at step $N+1$ is given by

$$W_i(N+1) = \frac{1}{2}[W_{i+1}(N) + W_{i-1}(N)]. \tag{7.23}$$

(Compare the relation (7.23) to the relation that you found in Problem 7.5d.) The survival probability S_N after N steps for a given configuration of traps is given exactly by

$$S_N = \frac{1}{w_0} \sum_i W_i(N), \tag{7.24}$$

where w_0 is the initial number of walkers and the sum is over all sites in the lattice. Explain the relation (7.24), and write a program that computes S_N using (7.23) and (7.24). Then obtain $\langle S_N \rangle$ by averaging over several configurations of traps. Choose $\rho = 0.5$ and determine S_N for $N = 32$, 64, 128, 512, and 1024. Choose periodic boundary conditions and as large a lattice as possible. How well can you estimate the exponent α? For comparison, Havlin et al. consider a lattice of $L = 50\,000$ and values of N up to 10^7.

One reason that random walks are very useful in simulating many physical processes and modeling many differential equations of physical interest is that their behavior is closely related to the solutions of the *diffusion* equation. The one-dimensional diffusion equation can be written as

$$\frac{\partial P(x,t)}{\partial t} = D \frac{\partial^2 P(x,t)}{\partial x^2}, \tag{7.25}$$

where D is the self-diffusion coefficient and $P(x,t)\,dx$ is the probability of a particle being in the interval between x and $x + dx$ at time t. In a typical application $P(x,t)$ might represent the concentration of ink molecules diffusing in a fluid. In three dimensions the second derivative

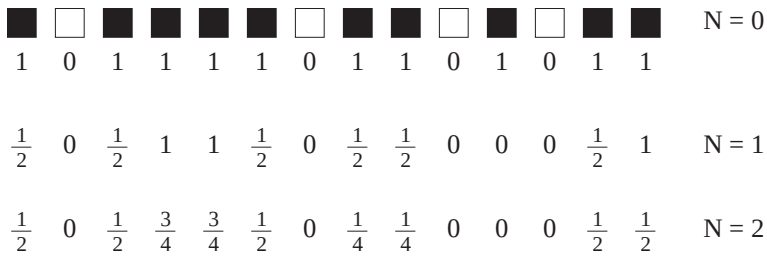

Figure 7.3: Example of the exact enumeration of walks on a given configuration of traps. The filled and empty squares denote non-trap and trap sites respectively. At step $N = 0$, a walker is placed at each regular site. The numbers at each site i represent the number of walkers W_i. Periodic boundary conditions are used. The initial number of walkers in this example is $w_0 = 10$. The mean survival probability at step $N = 1$ and $N = 2$ is found to be 0.6 and 0.475 respectively.

$\partial^2/\partial x^2$ is replaced by the Laplacian ∇^2. In Appendix 7B we show that the solution to the diffusion equation with the boundary condition $P(x = \pm\infty, t) = 0$ yields

$$\langle x_N \rangle = 0 \tag{7.26}$$

and

$$\langle x_N^2 \rangle = 2Dt. \tag{7.27}$$

If we compare the form of (7.8) with (7.27), we see that the random walk and the diffusion equation give the same time dependence if we identify t with $N\Delta t$ and $2D$ with $\ell^2/\Delta t$.

The relation of discrete random walks to the diffusion equation is an example of how we can approach many problems in two ways. The traditional way is to formulate the problem as a partial differential equation as in (7.25) and solve the equation by various numerical methods. One difficulty with this approach is the treatment of complicated boundary conditions. An alternative approach is to formulate the problem as a random walk. Later we will consider random walks in many contexts and find that it is straightforward to handle various boundary conditions (see for example, Section 10.5 and Chapter 16).

7.4 The Poisson Distribution and Nuclear Decay

As we have seen, we often can change variable names and do a seemingly different physical problem. Our goal in this section is to discuss the decay of unstable nuclei, but we first discuss a conceptually easier problem related to throwing darts at random. Related physical problems are the distribution of stars in the sky and the distribution of photons on a photographic plate.

Suppose we randomly throw $N = 100$ darts at a board that has been divided into $M = 1000$ equal size regions. (Usually our throws are not intended to be random, but here we consider random throws.) The probability that a dart hits a given region or cell in any one throw is $p = 1/M$. If we

count the number of darts in the different regions, we would find that most cells are empty, some cells have one dart, and other cells have more than one dart. What is the probability $P(n)$ that a given cell has n darts?

Problem 7.18. Throwing darts

Write a program that simulates the throwing of N darts at random into M cells in a dart board. Throwing a dart at random at the board is equivalent to choosing an integer at random between 1 and M. Determine $H(n)$, the number of cells with n darts. Average $H(n)$ over many trials, and then compute the distribution

$$P(n) = \frac{H(n)}{M}. \tag{7.28}$$

As an example, choose $N = 50$ and $M = 500$. Choose the number of trials to be sufficiently large so that you can determine the qualitative form of $P(n)$. What is $\langle n \rangle$?

In this case, the probability p that a dart lands in a given cell, is much less then unity. The conditions $N >> 1$ and $p << 1$ with $\langle n \rangle = Np$ fixed and the independence of the events (the presence of a dart in a particular cell) satisfy the requirements for a *Poisson distribution*, $P(n)$. This form of the Poisson distribution is

$$P(n) = \frac{\langle n \rangle^n}{n!} e^{-\langle n \rangle}, \tag{7.29}$$

where n is the number of darts in a given cell and $\langle n \rangle$ is the mean number, $\langle n \rangle = \sum_{n=0}^{\infty} nP(n)$. The upper limit of the sum should be N, but because $N >> 1$, we can take the upper limit to be ∞ when it is convenient.

Problem 7.19. Darts and the Poisson distribution

a. Write a program to compute $\sum_{n=0} P(n)$, $\sum_{n=0} nP(n)$, and $\sum_{n=0} n^2 P(n)$ using the form (7.29) for $P(n)$ and reasonable values of p and N. Verify that $P(n)$ in (7.29) is normalized. What is the value of σ for the Poisson distribution? Choose a value of the upper limit on the sum so that the various sums are unaffected.

b. Modify the program that you developed in Problem 7.18 to compute $\langle n \rangle$ as well as $P(n)$. Choose $N = 50$ and $M = 1000$ and use your measured value of $\langle n \rangle$ as input to compare o your computed values of $P(n)$ to the Poisson distribution. If time permits, use larger values of N and M.

c. Choose $N = 50$ and $M = 100$ and redo part (b). Are your results consistent with a Poisson distribution? What happens if $M = N = 50$?

Now that we are more familiar with the Poisson distribution, we consider the decay of radioactive nuclei. We know that a collection of radioactive nuclei will decay and there is no way to know a priori which nucleus will decay next. If all nuclei of a particular type are identical, why do they not all decay at the same time? The answer is based on the fundamental uncertainty inherent in the quantum description of matter at the microscopic level. In the following, we will see that a simple model of the decay process leads to an exponential decay law. This approach complements the continuum approach discussed in Section 3.8.

Because each nucleus is identical, we assume that during any time interval Δt, each nucleus has the same probability per unit time p of decaying. The basic algorithm is simple – choose an unstable nucleus and generate a random number r uniformly distributed in the unit interval $0 \leq r < 1$. If $r \leq p$, the unstable nucleus decays; otherwise, it does not. Each unstable nucleus is tested during each unit time interval. Note that for a system of unstable nuclei, there are many events that can happen during each time interval, for example, $0, 1, 2, \ldots, n$ nuclei can decay. In contrast, for the particles in the box problem, there is a probability of unity that a particle is moved from one side to the other side at each time interval. Remember that once a nucleus decays, it is no longer in the group of unstable nuclei that is tested at each time interval. Class `Nuclei` in Listing 7.5 implements the nuclear decay algorithm.

Listing 7.5: The `Nuclei` class.

```
/*
 * The org.opensourcephysics.sip.ch11 package contains classes for Chapter 7,
 * Random Processes, of the book Simulations in Physics.
 * Copyright (c) 2005 H. Gould, J. Tobochnik, and W. Christian.
 */
package org.opensourcephysics.sip.ch07;
/**
 * Nuclei allows unstable nuclei to decay
 *
 * @author Jan Tobochnik
 */

public class Nuclei {

        int n [];    // accumulated data on number of unstable nuclei, index is time
        int tMax; // maximum time to record data
        int n0;  // initial number of unstable nuclei
        double p; // decay probability

  /**
   * Initializes unstable nuclei array
   */

public void initialize () {
    n = new int[tMax+1];
}

  /**
   * Nuclei decay
   */

public void step() {
   n[0] += n0;
   int nUnstable = n0;
   for(int t = 0; t < tMax; t++) {
      for(int i = 0; i < nUnstable; i++)
         if(Math.random() < p) nUnstable--;
```

```
        n[t+1] += nUnstable;
      }
    }
  }
```

Problem 7.20. Monte Carlo simulation of nuclear decay

a. Write a target class that extends the `AbstractSimulation` class, simulates many trials, and plots the average number of unstable nuclei as a function of time. Assume that the time interval is one second. Choose `n0` $= 1000$ (the initial number of unstable nuclei), $p = 0.01$, and `tmax` $= 100$, and run for about 100 trials. Is your result for $N(t)$, the mean number of unstable nuclei at time t, consistent with the expected behavior, $N(t) = N(0)\, e^{-\lambda t}$ found in Section 3.8? What is the value of λ for this value of p?

b. There are a very large number of unstable nuclei in a typical radioactive source. We also know that over any reasonable time interval, only a relatively small number decay. Because $N >> 1$ and $p << 1$, we expect that $P(n)$, the probability that n nuclei decay during a specified time interval, is a Poisson distribution. Modify your target class so that it outputs the probability that n unstable nuclei decay during the first time interval. Choose `n0` $= 1000$, $p = 0.001$, `tmax` $= 1$, and average over 1000 traisl. What is the mean number $\langle n \rangle$ of nuclei that decay during this interval? What is the associated variance? Plot $P(n)$ versus n and compare your results to the Poisson distribution (7.29) with your measured value of $\langle n \rangle$ as input. Then consider $p = 0.02$.

c. Modify your target class so that it outputs the probability that n unstable nuclei decay during two time intervals. Choose `n0` $= 1000$, $p = 0.001$, and `tmax` $= 2$. Average over 1000 trials. Compare the probability you obtain with your results from part (b). How do your results change as the time interval becomes larger?

d. Increase p for fixed `n0` $= 1000$ and determine $P(n)$ for a given time interval. Estimate the values of p and n for which the Poisson distribution is no longer applicable.

e. Modify your program so that it flashes a small circle on the screen or makes a sound (like that of a Geiger counter) when a nucleus decays. You can have the computer make a beep by using the method `Toolkit.getDefaultToolkit().beep()` in `java.awt`. Choose the location of the small circle at random. Do a single run and describe the qualitative differences between the visual or audio patterns for the situations in parts (a)–(d)? Choose `n0` ≥ 5000. Such a visualization might be somewhat misleading on a serial computer because only one nuclei can be considered at a time. In contrast, for a real system, the nuclei can decay simultaneously. For what circumstances would your visualization or audio pattern be close to that of a real system? How can you improve the visualization to better approximate a real system?

7.5 Problems in Probability

Because many of the questions in this chapter can be answered by analytical methods, why bother to simulate these processes? The main reason is that it is simpler to introduce new methods in a

familiar context. Another reason is that if we change the nature of the random processes slightly, it often happens that it is difficult or impossible to obtain the answers by familiar methods. Still another reason is that writing a program and doing a simulation can aid your intuitive understanding of the system, especially if the problem involves the subtle concept of probability. Probability is an elusive concept in part because it cannot be measured at one time. To reinforce the importance of thinking about how to solve a problem on a computer, we suggest some problems in probability in the following. Does thinking about how to code these problems on a computer help you to find a pencil and paper solution?

Problem 7.21. Three boxes: stick or switch?

Suppose that there are three identical boxes, each with a lid. When you leave the room, a friend places a $10 bill in one of the boxes and closes the lid of each box. The friend knows the location of the $10 bill, but you do not. You then reenter the room and guess which one of the boxes has the $10 bill. As soon as you do, your friend opens the lid of a box that is empty. If you have chosen an empty box, your friend will open the lid of the other empty box. If you have chosen the right box, your friend will open the lid of one of the two empty boxes. You now have the opportunity to stay with your original choice or switch to the other unopened box. Suppose that you play this game many times and that each time you guess correctly, you keep the money. To maximize your winnings, should you maintain your initial choice or should you switch? Which strategy is better? Write a program to simulate this game and output the probability of winning for switching and for not switching. It is likely that before you finish your program, the correct strategy will become clear. To make your program more useful, consider four or five boxes.

Problem 7.22. Conditional probability

Suppose that many people in a community are tested at random for HIV. The accuracy of the test is 87% and the incidence of the disease in the general population, independent of any test, is 1%. If a person tests positive for HIV, what is the probability that this person really has HIV? Write a program to compute the probability. The answer can be found by using Bayes' theorem (cf. Bernardo and Smith). The answer is much less than 87%.

Problem 7.23. The roll of the dice

a. Write a program to compute the probability of obtaining at least one double six in twenty-four throws of a pair of die.

b. Suppose that two gamblers each begin with $100 in capital and on each throw of a coin, one gambler must win $1 and the other must lose $1. How long can they play on the average until the capital of the loser is exhausted? How long can they play if they each begin with $1000? Neither gambler is allowed to go into debt.

Problem 7.24. The Boys of Summer

Luck plays a large role in the outcome of any baseball season. The National League Central Division standings for 2004 are given in Table 7.1. Suppose that the teams remain unchanged and their probability of winning a particular game is given by their 2004 winning percentage. Do a simulation to determine the probability that the Cardinals would lead the division for another season. For simplicity, assume that the teams play only each other.

Team	Won	Lost	Percentage
St. Louis Cardinals	105	57	0.648
Houston Astros	92	70	0.568
Chicago Cubs	89	73	0.469
Pittsburgh Pirates	72	89	0.447
Cincinnati Reds	76	86	0.426
Milwaukee Brewers	67	94	0.416

Table 7.1: The National League Central standings for 2004.

Much of the present day motivation for the development of probability comes from science rather than from gambling. The next problem has much to do with statistical physics even though this application is not apparent.

Problem 7.25. Money exchange

Consider a two-dimensional plane that has been subdivided into cells, for example, a checkerboard. There can be an indefinite number of coins stacked on each cell. For simplicity, we initially assign one coin to each cell. The game proceeds as follows. Select two cells at random. If there is at least one coin on the first cell, move one coin to the second cell. If the first cell is empty, then do nothing. After many coin exchanges, how is the occupancy of the cells distributed? Are the coins uniformly distributed as in the initial state or are many cells empty? DWrite a Monte Carlo program to simulate this game and show the state of the cells visually. Consider a system with at least 16×16 cells. Plot the histogram $H(n)$ versus n, where $H(n)$ is the number of cells with n coins. Do your results change qualitatively if you consider bigger systems or begin with more coins on each cell?

Problem 7.26. Distribution of cooking times

An industrious physics student finds a job at a local fast food restaurant to help him pay his way through college. His task is to cook 20 hamburgers on a grill at any one time. When a hamburger is cooked, he is supposed to replace it with an uncooked hamburger. However, our physics major does not pay attention to whether the hamburger is cooked or not. His method is to choose a hamburger at random and replace it by a uncooked one and not bother to check whether the hamburger that he removes from the grill is ready or not. What is the distribution of cooking times of the hamburgers that he removes? To simplify the problem, assume that he replaces a random hamburger at regular intervals of thiry seconds and that there is an indefinite supply of uncooked hamburgers. Does the qualitative nature of the distribution change if he cooks 40 hamburgers at any one time?

7.6 Method of Least Squares

In Problem 7.20 we did a simulation of $N(t)$, the number of unstable nuclei at time t. Given the finite accuracy of our data, how do we know if our simulation results are consistent with the exponential relation between N and t? The approach that we have been using is to plot the

computed values of $\log N(t)$ as a function of t and to rely on our eye to help us draw the curve that best fits the data points. Such a visual approach works best when the curve is a straight line, that is, when the relation is linear. The advantages of this approach are that it is straightforward and allows us to see what we are doing. For example, if a data point is far from the straight line, or if there is a gap in the data, we will notice it easily. If the true analytical relation is not linear, it is likely that we will notice that the data points do not fit a simple straight line, but instead show curvature. If we blindly let a computer fit the data to a straight line, we might not notice that the fit is not very good unless we already have had much experience fitting data by hand. Finally, the visceral experience of fitting the data manually gives us some feeling for the nature of the data that might otherwise be missed. It usually is a good idea to plot some data in this way even though a computer can do it much faster.

Although the visual approach is simple, it does not yield precise results, and we also need to use more systematic fitting methods. The most common method for finding the best straight line fit to a series of measured points is called *linear regression* or *least squares*. Suppose we have n pairs of measurements $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ and that the errors are entirely in the values of y. For convenience, we assume that the uncertainties in $\{y_i\}$ all have the same magnitude. Our goal is to obtain the best fit to the linear function

$$y = mx + b. \tag{7.30}$$

The problem is to calculate the values of the parameters m and b for the best straight line through the n data points. The difference,

$$d_i = y_i - mx_i - b, \tag{7.31}$$

is a measure of the discrepancy in y_i. It is reasonable to assume that the best pair of values of m and b are those that minimize the quantity

$$\chi^2 = \sum_{i=1}^{n} (y_i - mx_i - b)^2. \tag{7.32}$$

Why should we minimize the sum of the squared differences between the experimental values, y_i, and the analytical values, $mx_i + b$, and not some other function of the differences? The justification is based on the assumption that if we did many simulations or measurements, then the values of d_i would be distributed according to the Gaussian distribution (see Problems 7.5 and 7.15). Based on this assumption, it can be shown that the values of m and b that minimize χ yield a set of values of $mx_i + b$ that are the *most probable* set of measurements that we would find based on the available information. This link to probability is the reason we have discussed least squares fits in this chapter, even though we will not explicitly use Monte Carlo methods here to show that the difference d_i is distributed according to a Gaussian distribution.

To minimize χ, we take the partial derivative of S with respect to b and m:

$$\frac{\partial \chi}{\partial m} = -2 \sum_{i=1}^{n} x_i (y_i - mx_i - b) = 0, \tag{7.33a}$$

$$\frac{\partial \chi}{\partial b} = -2 \sum_{i=1}^{n} (y_i - mx_i - b) = 0. \tag{7.33b}$$

From (7.33) we obtain two simultaneous equations:

$$m\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \tag{7.34a}$$

$$m\sum_{i=1}^{n} x_i + bn = \sum_{i=1}^{n} y_i. \tag{7.34b}$$

It is convenient to define the average quantities

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{7.35a}$$

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{7.35b}$$

$$\overline{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i, \tag{7.35c}$$

and rewrite (7.34) as

$$m\overline{x^2} + b\overline{x} = \overline{xy}, \tag{7.36a}$$

$$m\overline{x} + b = \overline{y}. \tag{7.36b}$$

The solution of (7.36) can be expressed as

$$m = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\sigma_x^2} \tag{7.37a}$$

$$b = \overline{y} - m\,\overline{x}, \tag{7.37b}$$

where

$$\sigma_x^2 = \overline{x^2} - \overline{x}^2. \tag{7.37c}$$

Equation (7.37) determines the slope m and the intercept b of the best straight line through the n data points. (Note that the average in the equations for the coefficients m and b are over the data points.)

As an example, consider the data shown in Table 7.2 for a one-dimensional random walk. To make the example more interesting, suppose that the walker takes steps of length 1 or 2 with equal probability. The direction of the step is random and $p = \frac{1}{2}$. As in Section 7.2, we assume that the mean square displacement $\langle x^2\rangle - \langle x\rangle^2$ obeys the general relation

$$\langle x^2\rangle - \langle x\rangle^2 = aN^{2\nu}, \tag{7.38}$$

with an unknown exponent ν. First we convert the nonlinear relation (7.38) to a linear relation by taking the logarithm of both sides:

$$\ln[\langle x^2\rangle - \langle x\rangle^2] = \ln a + 2\nu \ln N. \tag{7.39}$$

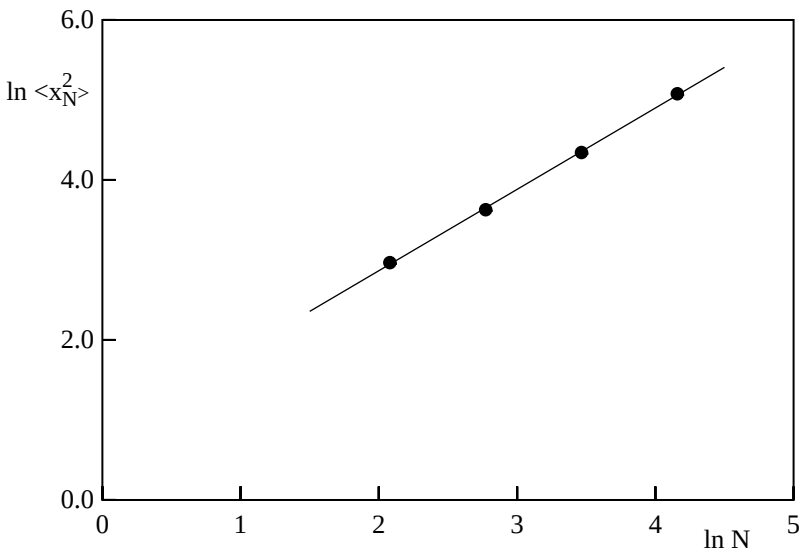

Figure 7.4: Plot of $\langle x^2\rangle - \langle x\rangle^2$ versus $\ln N$ for the data listed in Table 7.2. The straight line $y = 1.02x + 0.83$ through the points is found by minimizing the sum (7.32).

Note that the original fitting problem is nonlinear, that is, $\langle x^2\rangle - \langle x\rangle^2$ depends on N^ν rather than N. Often a problem that looks nonlinear can be turned into a linear problem by a change of variables.

If we take the logarithm of the data in Table 7.2 and use (7.37), we find that $m = 1.02$ and $b = 0.83$. Hence, we conclude from our limited data and the relation $2\nu = m$ that $\nu \approx 0.51$, which is consistent with the expected result $\nu = 1/2$. The values of $y = \ln[\langle x^2\rangle - \langle x\rangle^2]$ and $x = \ln N$ and the least squares fit are shown in Figure 7.4.

N	$\langle x^2\rangle - \langle x\rangle^2$
8	19.43
16	37.65
32	76.98
64	160.38

Table 7.2: Computed values of the mean square displacement $\langle x^2\rangle - \langle x\rangle^2$ as a function of the total number of steps N. The mean square displacement was averaged over 1000 trials. The one-dimensional random walker takes steps of length 1 or 2 with equal probability, and the direction of the step is random with $p = \frac{1}{2}$.

The least squares fitting procedure also allows us to estimate the uncertainty or the most probable error in m and b by analyzing the measurements themselves. The result of this analysis

is that the most probable error in m and b, σ_m and σ_b respectively, is given by

$$\sigma_m = \frac{1}{\sqrt{n}} \frac{\Delta}{\sigma_x} \tag{7.40a}$$

$$\sigma_b = \frac{1}{\sqrt{n}} \frac{\left(\overline{x^2}\right)^{1/2} \Delta}{\sigma_x}, \tag{7.40b}$$

where

$$\Delta^2 = \frac{1}{n-2} \sum_{i=1}^{n} d_i^2, \tag{7.41}$$

and d_i is given by (7.31).

Because there are n data points, we might have guessed that n rather than $n-2$ would be present in the denominator of (7.41). The reason for the factor of $n-2$ is related to the fact that to determine Δ, we first need to calculate *two* quantities m and b, leaving only $n-2$ independent degrees of freedom. To see that the $n-2$ factor is reasonable, consider the special case of $n=2$. In this case we can find a line that passes exactly through the two data points, but we cannot deduce anything about the reliability of the set of measurements because the fit always is exact. If we use (7.41), we see that both the numerator and denominator would be zero, and hence Δ is undetermined. If a factor of n rather than $n-2$ appeared in (7.41), we would conclude that $\Delta = 0/2 = 0$, an absurd conclusion. Usually $n >> 1$, and the difference between n and $n-2$ is negligible.

For our example, $\Delta = 0.03$, $\sigma_b = 0.07$, and $\sigma_m = 0.02$. The uncertainties δm and $\delta\nu$ are related by $2\delta\nu = \delta m$. Because $\delta m = \sigma_m$, we conclude that our best estimate for ν is $\nu = 0.51 \pm 0.01$.

If the values of y_i have different uncertainties σ_i, then the data points are weighted by the quantity $w_i = 1/\sigma_i^2$. In this case it is reasonable to minimize the quantity

$$\chi^2 = \sum_{i=1}^{n} w_i (y_i - m x_i - b)^2. \tag{7.42}$$

The resulting expressions in (7.37) for m and b are unchanged if we generalize the definition of the averages to be

$$\overline{f} = \frac{1}{n\overline{w}} \sum_{i=1}^{n} w_i f_i, \tag{7.43}$$

where

$$\overline{w} = \frac{1}{n} \sum_{i=1}^{n} w_i. \tag{7.44}$$

Problem 7.27. Example of least squares fit

a. Write a program to find the least squares fit for a set of data. As a check on your program, compute the most probable values of m and b for the data shown in Table 7.2.

b. Modify the random walk program so that steps of length 1 and 2 are taken with equal probability. Use at least 10 000 trials and do a least squares fit to $\langle x^2\rangle - \langle x\rangle^2$ as done in the text. Is your most probable estimate for ν closer to $\nu = 1/2$?

For simple random walk problems the relation $\langle x^2\rangle - \langle x\rangle^2 = aN^\nu$ holds for all N. However, in many random walk problems a power law relation between $\langle x^2\rangle - \langle x\rangle^2$ and N holds only asymptotically for large N, and hence we should use only the larger values of N to estimate the slope. Also, because we are finding the best fit with the logarithm of the independent variable N, we need to give equal weight to all intervals of $\ln N$. In the above example, we used $N = 8$, 16, 32, and 64, so that the values of $\ln N$ are equally spaced.

7.7 Applications to Polymers

Random walk models play an important role in polymer physics (cf. de Gennes). A polymer consists of N repeat units (monomers) with N very large ($N \sim 10^3 - 10^5$). For example, polyethylene can be represented as $\cdots{-}\mathrm{CH_2}{-}\mathrm{CH_2}{-}CH_2{-}\cdots$. The detailed structure of the polymer is important for many practical applications. For example, if we wish to improve the fabrication of rubber, a good understanding of the local motions of the monomers in the rubber chain is essential. However, if we are interested in the global properties of the polymer, the details of the chain structure can be ignored.

Let us consider a familiar example of a polymer chain in a good solvent: a noodle in warm water. A short time after we place a noodle in warm water, the noodle becomes flexible, and it neither collapses into a little ball or becomes fully stretched. Instead, it adopts a random structure as shown schematically in Figure 7.5. If we do not add too many noodles, we can say that the noodles behave as a dilute solution of polymer chains in a good solvent. The dilute nature of the solution implies that we can ignore entanglement effects of the noodles and consider each noodle individually. The presence of a good solvent implies that the polymers can move freely and adopt many different configurations.

A fundamental geometrical property that characterizes a polymer in a good solvent is the mean square end-to-end distance $\langle R_N^2\rangle$, where N is the number of monomers. (For simplicity, we will frequently omit the subscript N in the following.) It is known that for a dilute solution of polymer chains in a good solvent, the asymptotic dependence of $\langle R^2\rangle$ is given by (7.13) with $\nu \approx 0.5874$ in three dimensions. However, if we were to ignore the interactions of the monomers, the simple random walk model would yield $\nu = \frac{1}{2}$, independent of the dimension and symmetry of the lattice. Because this result for ν does not agree with experiment ($\nu > 1/2$), we know that we are overlooking an important physical feature of polymers.

We now discuss a random walk that incorporates the global features of dilute linear polymers in solution. We already have introduced a model of a polymer chain consisting of straight line segments of the same size joined together at random angles (see Problem 7.13). A further idealization is to place the polymer chain on a lattice (see Figure 7.5). A more realistic model of linear polymers accounts for its most important physical feature, that is, two monomers cannot occupy the same spatial position. This constraint is known as the *excluded volume* condition, which is ignored in a simple random walk. A well known lattice model for a linear polymer chain that incorporates this constraint is known as the *self-avoiding* walk (SAW). This model consists of the set of all

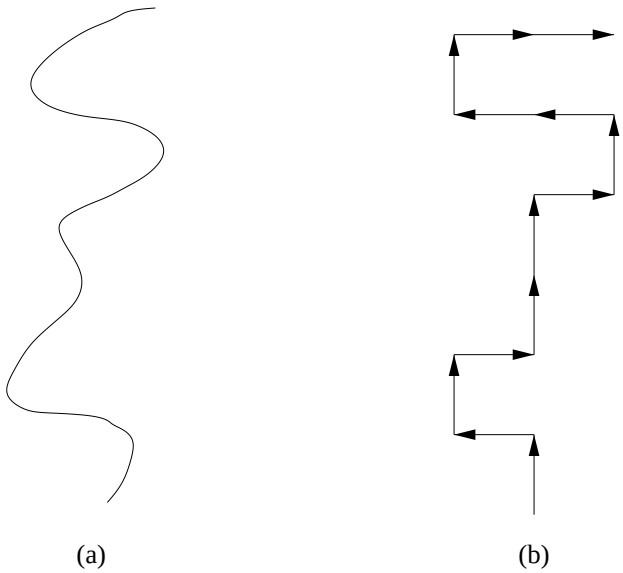

Figure 7.5: (a) Schematic illustration of a linear polymer in a good solvent. (b) Example of the corresponding self-avoiding walk on a square lattice.

N step walks starting from the origin subject to the global constraint that no lattice site can be visited more than once in each walk; this constraint accounts for the excluded volume condition. Self-avoiding walks have many applications in the sciences, such as the physics of magnetic materials and the study of phase transitions, and they are of interest as purely mathematical objects. Many of the obvious questions have resisted rigorous analysis, and exact enumeration and Monte Carlo simulation have played an important role in our current understanding. The result for ν in two dimensions for the SAW is known to be exactly $\nu = 3/4$; in three dimensions the exponent $\nu \approx 0.5874$ in agreement with experiment. The proportionality constant in (7.13) depends on the structure of the monomers and on the solvent. In contrast, the exponent ν is independent of these details and depends only the spatial dimension.

We consider Monte Carlo simulations of the self-avoiding walk in two dimensions in Problems 7.28 and 7.30. Another algorithm for the self-avoiding walk is considered in Project 7.41.

Problem 7.28. The two-dimensional self-avoiding walk

a. Consider the self-avoiding walk on the square lattice. Choose an arbitrary site as the origin and assume that the first step is "up." The walks generated by the three other possible initial steps only differ by a rotation of the whole lattice and do not have to be considered explicitly. The second step can be in three possible directions because of the constraint that the walk cannot return to the origin. To obtain unbiased results, we generate a random number to choose one of the three directions. Successive steps are generated in the same way. Unfortunately, the walk will not continue indefinitely. The difficulty is that to obtain unbiased results, we must choose at random one of the three steps, even though one or more of these steps might lead to

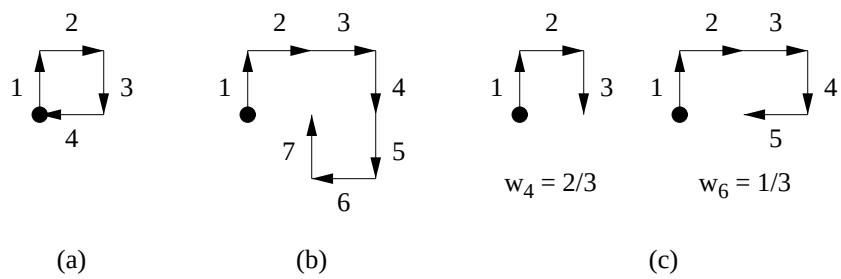

Figure 7.6: Examples of self-avoiding walks on the square lattice. The origin is denoted by a filled circle. (a) A $N = 3$ walk. The fourth step shown is forbidden. (b) A $N = 7$ walk that leads to a self-intersection at the next step; the weight of the $N = 8$ walk is zero. (c) Examples of the weights of walks in the enrichment method.

a self-intersection. If the next step does lead to a self-intersection, the walk must be terminated to keep the statistics unbiased. An example for a three step walk is shown in Figure 7.6a. The next step leads to a self-intersection and violates the constraint. In this case we must start a new walk at the origin.

Write a program that implements this algorithm and record the fraction $f(N)$ of successful attempts at constructing polymer chains with N total monomers. It is convenient to represent the lattice as a two-dimensional array so that you can record the sites that already have been visited. What is the qualitative dependence of $f(N)$ on N? What is the maximum value of N that you can reasonably consider?

b. One of the quantities of interest is the end-to-end distance, that is, the distance from the origin to the end of the polymer. Determine the mean square end-to-end distance $\langle R_N^2 \rangle$ for the various values of N that you can reasonably consider with this sampling method.

The disadvantage of the straightforward sampling method in Problem 7.28 is that it becomes very inefficient for long chains, that is, the fraction of successful attempts decreases exponentially fast. To overcome this attrition, several "enrichment" techniques have been developed. We first discuss a relatively simple procedure proposed by Rosenbluth and Rosenbluth in which each walk of N steps is associated with a weighting function $W(N)$. Because the first step to the north is always possible, we have $W(1) = 1$. In order that all allowed configurations of a given N are counted equally, the weights $W(N)$ for $N > 1$ are determined according to the following possibilities:

1. All three possible steps violate the self-intersection constraint (see Figure 7.6b). The walk is terminated with a weight $W(N) = 0$, and a new walk is generated at the origin.

2. All three steps are possible and $W(N) = W(N-1)$.

3. Only m steps are possible with $1 \leq m < 3$ (see Figure 7.6c). In this case $W(N) = (m/3)W(N-1)$, and one of the m possible steps is chosen at random.

The correct (unbiased) value of $\langle R_N^2 \rangle$ is obtained by weighting $R_{N,i}^2$, the value of R_N^2 found in the ith trial, by the value of $W_i(N)$, the weight found for the particular walk. Hence we write

$$\langle R_N^2 \rangle = \frac{\sum_i W_i(N) R_{N,i}^2}{\sum_i W_i(N)}, \tag{7.45}$$

where the sum is over all trials.

Problem 7.29. Rosenbluth and Rosenbluth enrichment method

Incorporate the Rosenbluth method into your Monte Carlo program, and calculate $\langle R_N^2 \rangle$ for $N = 4, 8, 16$, and 32. Estimate the exponent ν from a log-log plot of $\langle R_N^2 \rangle$ versus N. Can you distinguish your estimate for ν from its random walk value $\nu = \frac{1}{2}$?

The Rosenbluth and Rosenbluth procedure is not particularly efficient because many walks still die out, and thus we do not obtain many walkers for large N. Grassberger improved this algorithm by increasing the population of walkers with high weights and reducing the population of walkers with low weights. The idea is that if $W_i(N)$ is above a certain threshold, we add a new walker and give the new and old walker half of the original weight. If $W_i(N)$ is below a certain threshold, then we eliminate half of the walkers with weights below this threshold (for example, every second walker) and double the weights of the remaining half. It is a good idea to adjust the thresholds as the simulation runs in order to maintain a relatively constant number of walkers.

More recently Prelberg and Krawczyk made a further improvement in the Rosenbluth and Rosenbluth enrichment method so that there is no need for the user to provide a threshold value. After each step the average weight of the walkers, $\langle W_i(N) \rangle$ is computed and the ratio $r = W_i(N)/\langle W_i(N) \rangle$ is used to determine whether to add walkers (enrichment) or eliminate walkers (pruning). If $r > 1$, then $c = \min(r, m)$ copies of the walker are made each with weight $W_i(N)/c$. If $r < 1$, then remove this walker with probability $1 - r$. This algorithm leads to an approximately constant number of walkers and is related to the flat histogram method which we will discuss in Problem 15.30.

One of the more efficient enrichment algorithms is the "reptation" method (see Wall and Mandel). For simplicity, consider a model polymer chain in which all bond angles are $\pm 90°$. As an example of this model, the five independent $N = 5$ polymer chains are shown in Figure 7.7. (Other chains differ only by a rotation or a reflection.) The reptation method can be stated as follows:

1. Choose a chain at random and remove the tail link.
2. Attempt to add a link to the head of the chain. There is a maximum of two directions in which the new head link can be added.
3. If the attempt violates the self-intersection constraint, return to the original chain and interchange the head and tail. Include the chain in the statistical sample.

The above steps are repeated many times to obtain an estimate of $\langle R_N^2 \rangle$.

As an example of the reptation method, sconsider chain a of Figure 7.7. A new link can be added in two directions (see Figure 7.8a), so that on the average we find, $a \to \frac{1}{2}c + \frac{1}{2}d$. In contrast,

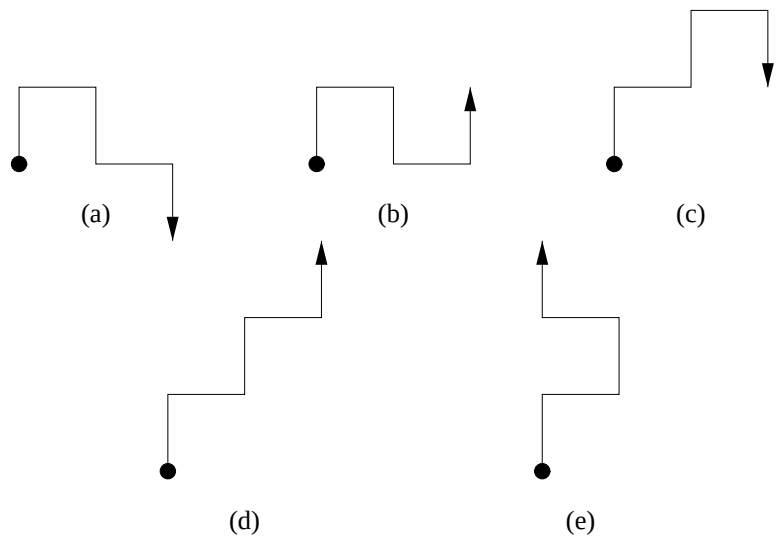

Figure 7.7: The five independent possible walks of $N = 5$ steps on a square lattice with $\pm 90°$ bond angles. The tail and head of each walk are denoted by a circle and arrow respectively.

a link can be added to chain b in only one direction, and we obtain $b \to \frac{1}{2}e + \frac{1}{2}b$, where the tail and head of chain b have been interchanged (see Figure 7.8b). Confirm that $c \to \frac{1}{2}e + \frac{1}{2}a$, $d \to \frac{1}{2}c + \frac{1}{2}d$, and $e \to \frac{1}{2}a + \frac{1}{2}b$, and that all five chains are equally probable. That is, the transformations in the reptation method preserve the proper statistical weights of the chains without attrition. There is just one problem: unless we begin with a double ended "cul-de-sac" configuration such as shown in Figure 7.9, we will never obtain such a configuration using the above transformation. Hence, the reptation method introduces a small statistical bias, and the calculated mean end-to-end distance will be slightly larger than if all configurations were considered. However, the probability of such trapped configurations is very small, and the bias can be neglected for most purposes.

***Problem 7.30.** The reptation method

Adopt the $\pm 90°$ bond angle restriction and calculate by hand the exact value of $\langle R_N^2 \rangle$ for $N = 5$. Then write a Monte Carlo program that implements the reptation method. Generate one walk of $N = 5$ and use the reptation method to generate a statistical sample of chains. As a check on your Monte Carlo program, compute $\langle R_N^2 \rangle$ for $N = 5$ and compare your result with the exact result. Then extend your Monte Carlo computations of $\langle R_N^2 \rangle$ to larger N. Modify the reptation model so that the bond angle also can be $180°$. This modification leads to a maximum of three directions for a new bond. Compare the results of the two models.

In principle, the dynamics of a polymer chain undergoing collisions with solvent molecules can be simulated by using a molecular dynamics method. However, in practice only relatively small chains can be simulated in this way. An alternative approach is to use a Monte Carlo model that simplifies the effect of the random collisions of the solvent molecules with the atoms of the chain. Most of these models (cf. Verdier and Stockmayer) consider the chain to be composed of

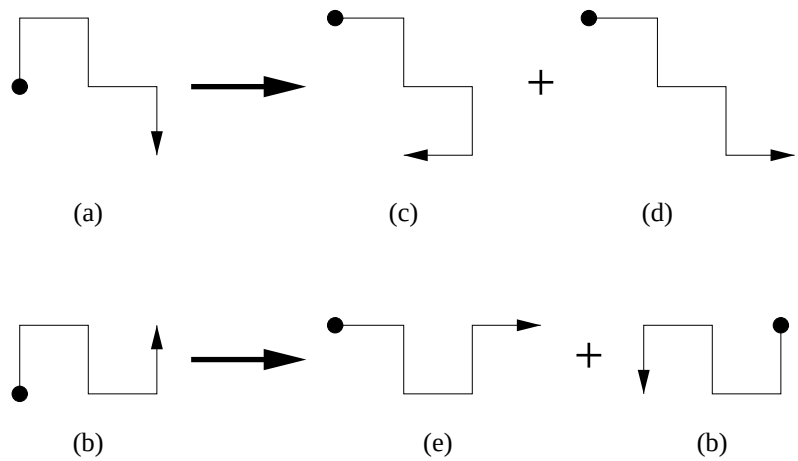

Figure 7.8: The possible transformations of chains a and b. One of the two possible transformations of chain b violates the self-intersection restriction and the head and tail are interchanged.

beads connected by bonds and restrict the motion of the beads to a lattice. For simplicity, we assume that the bond angles can be either $\pm 90°$ or $180°$. The idea is to begin with an allowed configuration of N beads ($N-1$ bonds). A possible starting configuration can be generated by taking successive steps in the positive y direction and positive x directions. The dynamics of the Verdier-Stockmayer algorithm is summarized by the following steps.

1. Select at random a bead (occupied site) on the polymer chain. If the bead is not an end site, then the bead can move to a nearest neighbor site of another bead if this site is empty and if the new angle between adjacent bonds is either $\pm 90°$ or $180°$. For example, bead 4 in Figure 7.10 can move to position $4'$ while bead 3 cannot move if selected. That is, a selected bead can move to a diagonally opposite unoccupied site only if the two bonds to which it is attached are mutually perpendicular.

2. If the selected bead is an end site, move it to one of two (maximum) possible unoccupied sites so that the bond to which it is connected changes its orientation by $\pm 90°$ (see Figure 7.10).

3. If the selected bead cannot move, retain the previous configuration.

The physical quantities of interest include $\langle R_N^2 \rangle$ and the mean square displacement of the center of mass of the chain $\langle r_{\rm cm}^2(N) \rangle = \langle x_{\rm cm}^2(N) \rangle - \langle x_{\rm cm}(N) \rangle^2 + \langle y_{\rm cm}^2(N) \rangle - \langle y_{\rm cm}(N) \rangle^2$, where $x_{\rm cm}$ and $y_{\rm cm}$ are the coordinates of the center of mass. The unit of time is the number of Monte Carlo steps per bead; in one Monte Carlo step per bead each bead has one chance on the average to move to a different site.

Problem 7.31. The dynamics of polymers in a dilute solution

a. Consider a two-dimensional lattice and compute $\langle R^2 \rangle$ and $\langle r_{\rm cm}^2 \rangle$ for various values of N. How do these quantities depend on N? (The first published results for three dimensions were limited

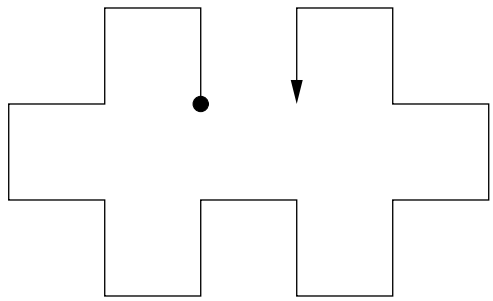

Figure 7.9: Example of a double cul-de-sac configuration for the self-avoiding walk that cannot be obtained by the reptation method.

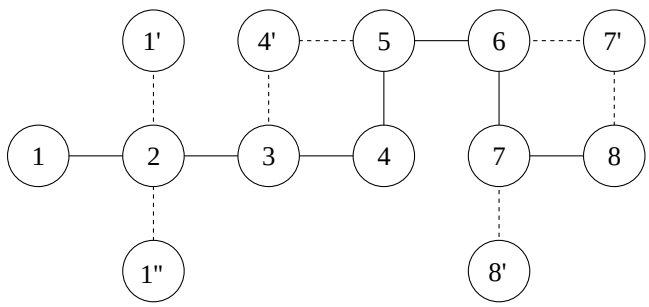

Figure 7.10: Examples of possible moves of the simple polymer dynamics model considered in Problem 7.31. For this configuration beads 2, 3, 5, and 6 cannot move, while beads 1, 4, 7, and 8 can move to the positions shown if they are selected. Only one bead can move at a time. This figure is adopted from the article by Verdier and Stockmayer.

to 32 Monte Carlo steps per bead for $N = 8, 16$, and 32 and only 8 Monte Carlo steps per bead for $N = 64$.) Also compute the probability density $P(R)$ that the end-to-end distance is R. How does this probability compare to a Gaussian distribution?

b.* We know that two configurations are strongly correlated if they differ by only the position of one bead. Hence, it would be a waste of computer time to measure the end-to-end distance and the position of the center of mass after every single move. Ideally, we wish to compute these quantities for configurations that are approximately statistically independent. Because we do not know *a priori* the mean number of Monte Carlo steps per bead needed to obtain configurations that are statistically independent, it is a good idea to estimate this time in our preliminary calculations. The correlation time τ, is the time needed to obtain statistically independent configurations. This time can be obtained by computing the equilibrium averaged

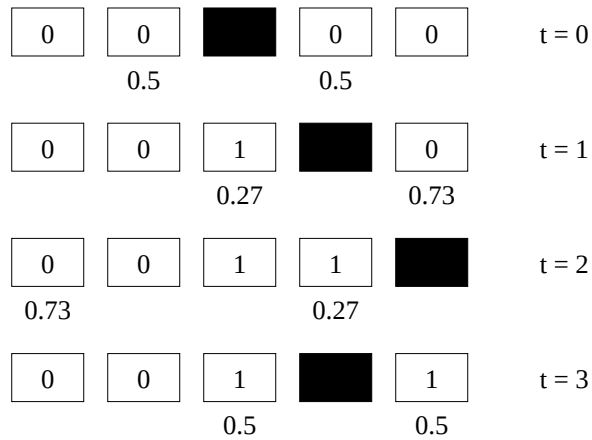

Figure 7.11: Example of the evolution of the true self-avoiding walk with $g = 1$ (see Figure 7.47). The shaded site represents the location of the walker at time t. The number of visits to each site are given within each site and the probability of a step to a nearest neighbor site is given below it. Note the use of periodic boundary conditions.

autocorrelation function for a chain of fixed N:

$$C_N(t) = \frac{\langle R_N^2(t'+t)R_N^2(t')\rangle - \langle R_N^2\rangle^2}{\langle R_N^4\rangle - \langle R_N^2\rangle^2}. \tag{7.46}$$

$C_N(t)$ is defined so that $C_N(t=0) = 1$ and $C_N(t) = 0$ if the configurations are not correlated. Because the configurations will become uncorrelated if the time t between the configurations is sufficiently long, we expect that $C_N(t) \to 0$ for $t >> 1$. We expect that $C(t) \sim e^{-t/\tau}$, that is, $C(t)$ decays exponentially with a decay or correlation time τ. Estimate τ from a plot of $\ln C(t)$ versus t. Another way of estimating τ is from the integral $\int_0^\infty dt\, C(t)$, where $C(t)$ is normalized so that $C(0) = 1$. (Because we determine $C(t)$ at discrete values of t, this integral is actually a sum.) How do your two estimates of τ compare? A more detailed discussion of the estimation of correlation times can be found in Section 15.7.

Another type of random walk that is less constrained than the self-avoiding random walk is the "true" self-avoiding walk (TSAW). The TSAW describes the path of a random walker that avoids visiting a lattice site with a probability that is a function of the number of times the site has been visited already. This constraint leads to a reduced excluded volume interaction in comparison to the usual self-avoiding walk.

Problem 7.32. The true self-avoiding walk in one dimension

In one dimension the true self-avoiding walk corresponds to a walker who can jump to one of its two nearest neighbors with a probability that depends on the number of times these neighbors already have been visited. Suppose that the walker is at site i at step t. The probability that at

time $t+1$, the walker will jump to site $i+1$ is given by

$$p_{i+1} = \frac{e^{-gn_{i+1}}}{e^{-gn_{i+1}} + e^{-gn_{i-1}}}, \tag{7.47}$$

where $n_{i\pm 1}$ is the number of times that the walker has already visited site $i \pm 1$. The probability of a jump to site $i-1$ is $p_{i-1} = 1 - p_{i+1}$. The parameter g ($g > 0$) is a measure of the "desire" of the path to avoid itself. The first few steps of a typical true self-avoiding walk walk are shown in Figure 7.11. The main quantity of interest is the exponent ν. We know that $g = 0$ corresponds to the usual random walk with $\nu = \frac{1}{2}$ and that the limit $g \to \infty$ corresponds to the self-avoiding walk. What is the value of ν for a self-avoiding walk in one dimension? Is the value of ν for any finite value of g different than these two limiting cases?

Write a program to do a Monte Carlo simulation of the true self-avoiding walk in one dimension. Use an array to record the number of visits to every site. At each step calculate the probability p of a jump to the right. Generate a random number r and compare it to p. If $r \leq p$, move the walker to the right; otherwise move the walker to the left. Compute $\langle \Delta x_N^2 \rangle$, where x is the distance of the walker from the origin, as a function of the number of steps N. Make a log-log plot of $\langle \Delta x_N^2 \rangle$ versus N and estimate ν. Can you distinguish ν from its random walk and self-avoiding walk values? Reasonable choices of parameters are $g = 0.1$ and $N \sim 10^3$. Averages over 10^3 trials yield qualitative results. For comparison, published results are for $N \sim 10^4$ and for 10^3 trials; extended results for $g = 2$ are given for $N = 2 \times 10^5$ and 10^4 trials (see Bernasconi and Pietronero).

7.8 Diffusion Controlled Chemical Reactions

Imagine a system containing particles of a single species A. The particles diffuse, and when two particles "collide," a reaction occurs such that the two combine to form an inert species which is no longer involved in the reaction. We can represent this chemical reaction as

$$A + A \to 0. \tag{7.48}$$

If we ignore the spatial distribution of the particles, we can describe the kinetics by a simple rate equation:

$$\frac{dA(t)}{dt} = -kA^2(t), \tag{7.49}$$

where A is the concentration of A particles at time t and k is the rate constant. (In the chemical kinetics literature it is traditional to use the term concentration rather than the number density.) For simplicity, we assume that all reactants are entered into the system at $t = 0$ and that no reactants are added later (the system is closed). It is easy to show that the solution of the first-order differential equation (7.49) is

$$A(t) = \frac{1}{kt + 1/A(0)}. \tag{7.50}$$

Hence, $A(t) \sim t^{-1}$ in the limit of long times.

Another interesting case is the bimolecular reaction

$$A + B \rightarrow 0. \tag{7.51}$$

If we neglect spatial fluctuations in the concentration as before (this neglect yields what is known as a mean-field approximation), we can write the corresponding rate equation as

$$\frac{dA(t)}{dt} = \frac{dB(t)}{dt} = -kA(t)B(t). \tag{7.52}$$

We also have that

$$A(t) - B(t) = \text{constant}, \tag{7.53}$$

because each reaction leaves the difference between the concentration of A and B particles unchanged. For the special case of equal initial concentrations, the solution of (7.52) with (7.53) is the same as (7.50). What is the solution for the case $A(0) \neq B(0)$?

The above derivation of the time dependence of A for the kinetics of the one and two species annihilation process is straightforward, but is based on the assumption that the particles are distributed uniformly. In the following two problems, we simulate the kinetics of these processes and test this assumption.

Problem 7.33. Diffusion controlled chemical reactions in one dimension

a. Assume that N particles do a random walk on a one-dimensional lattice of length L with periodic boundary conditions. Every particle moves once in one unit of time. It is convenient to use the array `site[j]` to records the label of the particle, if any, at site `j`. Because we are interested in the long time behavior of the system when the concentration $A = N/L$ of particles is small, it is efficient to also maintain an array of particle positions, `x[i]` such that `site[x(i]]` = `i`. For example, if particle 5 is located at site 12, then `x[5]` = 12 and `site[12]` = 5. We also need an array, `newSite`, to maintain the new positions of the walkers as they are moved one at a time. If two walkers land on the same position `k`, then we set `newSite[k]` $= -1$, and the value of `x[i]` for these two walkers to -1. The number -1 indicates that no particle exists at the site. After all the walkers have moved, we let `site` = `newSite` for all sites, and remove all the reacting particles in `x` that have values equal to -1. This operation can be accomplished by replacing any reacting particle in `x` by the last particle in the array. Begin with all sites occupied, $A(t = 0) = 1$.

b. Make a log-log plot of the quantity $A(t)^{-1} - A(0)^{-1}$ versus the time t. (The initial concentration is $A(t = 0) = 1$.) The times should be separated by exponential intervals so that your data is equally spaced on a logarithmic plot. For example, you might include data with times equal to 2^p, with $p = 1, 2, 3, \ldots$ Does your log-log plot yield a straight line in the limit of long times? If so, calculate its slope. Is the mean-field approximation for $A(t)$ valid in one dimension? You can obtain crude results for small lattices of order $L = 100$ and times of order $t = 10^2$. To obtain results to within10%, you need lattices of order $L = 10^4$ and times of order $t = 2^{13}$.

c. More insight into the origin of the time dependence of $A(t)$ can be gained from the behavior of the quantity $P(r, t)$, the probability that the nearest neighbor distance is r at time t. The nearest

neighbor distance of a given particle is defined as the minimum distance between it and all other particles. The distribution of these distances changes dramatically as the reaction proceeds, and this change can give information about the reaction mechanism. Place the particles at random on a one-dimensional lattice and verify that the most probable nearest neighbor distance is $r = 1$ (one lattice constant) for all concentrations. (This result is true in any dimension.) Then verify that the distribution of nearest neighbor distances on a $d = 1$ lattice is given by

$$P(r, t = 0) = 2A\, e^{-2A(r-1)}. \qquad \text{(random distribution)} \tag{7.54}$$

Is the form (7.54) properly normalized? Start with $A(t = 0) = 0.1$ and find $P(r, t)$ for $t = 10$, 100, and 1000. Average over all particles. How does $P(r, t)$ change as the reaction proceeds? Does it retain the same form as the concentration decreases?

d.* Compute the quantity $D(t)$, the number of *distinct* sites visited by an individual walker. How does the time dependence of $D(t)$ compare to the computed time dependence of $A(t)^{-1} - 1$?

e.* Write a program to simulate the $A + B = 0$ reaction. For simplicity, assume that multiple occupancy of the same site is not allowed, for example, an A particle cannot jump to a site already occupied by an A particle. The easiest procedure is to allow a walker to choose one of its nearest neighbor sites at random, but to not move the walker if the chosen site is already occupied by a particle of the same type. If the site is occupied by a walker of another type, then the pair of reacting particles is annihilated. Keep separate arrays for the A and B particles, with the value of the array denoting the label of the particle as before. An easy way to distinguish A and B walkers is to make the array element `site(k)` positive if the site is occupied by an A particle and negative if the site is occupied by a B particle. Start with equal concentrations of A and B particles and occupy the sites at random. Some of the interesting questions are similar to those that we posed in parts (b)–(d). Color code the particles and observe what happens to the relative positions of the particles.

***Problem 7.34.** Reaction diffusion in two dimensions

a. Do a similar simulation as in Problem 7.33 on a two-dimensional lattice for the reaction $A + A \to 0$. In this case it is convenient to have one array for each dimension, for example, `siteX` and `siteY`. Set $A(t = 0) = 1$, and choose $L = 50$. Show the configuration of walkers after each Monte Carlo step per walker. Describe the geometry of the clusters of particles as the diffusion process proceeds. Are the particles uniformly distributed throughout the lattice for all times? Calculate $A(t)$ and compare your results for $A(t)^{-1} - A(0)^{-1}$ to the t-dependence of $D(t)$, the number of distinct lattice sites that are visited in time t. ($D(t) \sim t/\log t$ for two dimensions.) How well do the slopes compare? Do a similar simulation with $A(t = 0) = 0.01$. What slope do you obtain in this case? What can you conclude about the initial density dependence? Is the mean-field approximation valid in this case? If time permits, do a similar simulation in three dimensions.

b. Begin with A and B type random walkers initially segregated on the left and right halves (in the x direction) of a square lattice. The process $A + B \to C$ exhibits a reaction front where the production of particles of type C is nonzero. Some of the quantities of interest are the time dependence of the mean position $\overline{x}(t)$ and the width $w(t)$ of the reaction front. The rules of this

process are the same as in part (a) except that a particle of type C is added to a site when a reaction occurs. A particular site can be occupied by one particle of type A or type B as well as any number of particles of type C. If $n(x,t)$ is the number of particles of type C at a distance x from the initial boundary of the reactants, then $x(t)$ and $w(t)$ can be written as

$$\overline{x}(t) = \frac{\sum_x x\, n(x,t)}{\sum_x n(x,t)} \tag{7.55}$$

$$w^2(t) = \frac{\sum_x \left[x - \overline{x}(t)\right]^2 n(x,t)}{\sum_x n(x,t)}. \tag{7.56}$$

Choose lattice sizes of order 100×100, and average over at least 10 trials. The fluctuations in $x(t)$ and $w(t)$ can be reduced by averaging $n(x,t)$ over the order of 100 time units centered about t. More details can be found in Jiang and Ebner.

7.9 Random Number Sequences

So far we have used the random number generator supplied with Java to generate the desired random numbers for our Monte Carlo applications. In principle, we might have generated these numbers from a random physical process, such as the decay of radioactive nuclei or the thermal noise from a semiconductor device. In practice, random number sequences are generated from a physical process only for specific purposes such as a lottery. Although we could store the outcome of a random physical process so that the random number sequence would be both truly random and reproducible, such a method would usually be inconvenient and inefficient in part because we often require very long sequences. There are companies that sell CDs and DVDs with many millions of *genuine* random numbers. However, in practice we use a digital computer, a deterministic machine, to generate sequences of *pseudorandom* numbers. Although these sequences cannot be truly random, such a distinction is unimportant if the sequence satisfies all our criteria for randomness. It is common to refer to random number generators even though we really mean pseudorandom number generators.

Most random number generators yield a sequence in which each number is used to find the succeeding one according to a well defined algorithm. The most important features of a desirable random number generator are that its sequence satisfies the known statistical tests for randomness, which we will explore in the following problems. We also want the generator to be efficient and machine independent. and the sequence to be reproducible.

The most widely used random number generator is based on the *linear congruential* method. That is, given the *seed* x_0, each number in the sequence is determined by the one-dimensional map

$$x_n = (a x_{n-1} + c) \bmod \mathrm{m}, \tag{7.57}$$

where a, c, and m as well as x_n are integers. The notation $y = z \bmod \mathrm{m}$ means that m is subtracted from z until $0 \le y < m$. The map (7.57) is characterized by three parameters, the *multiplier* a, the *increment* c, and the *modulus* m. Because m is the largest integer generated by (7.57), the maximum possible *period* is m.

In general, the period depends on all three parameters. For example, if $a = 3$, $c = 4$, $m = 32$, and $x_0 = 1$, the sequence generated by (7.57) is 1, 7, 25, 15, 17, 23, 9, 31, 1, 7, 25, ... and the

period is 8 rather than the maximum possible value of $m = 32$. If we are careful to choose a, c, and m such that the maximum period is obtained, then all possible integers between 0 and $m-1$ would occur in the sequence. Because we usually wish to have random numbers r in the unit interval $0 \leq r < 1$ rather than random integers, random number generators usually return the ratio x_n/m which is always less than unity. As the above example illustrates, m, a, and c must be chosen carefully to achieve optimum results. Several rules have been developed (see Knuth) to obtain the longest period. One advantage of the linear congruential method is that it is very fast. Some of the properties of the linear congruential method are explored in Problem 7.35.

Another popular random number generator is the *generalized feedback shift register* method which uses bit manipulation. Every integer is represented as a series of 1's and 0's called bits. These bits can be shuffled by using the bitwise `exclusive or` operator $\oplus$ (xor) defined by $a \oplus b = 1$ if the bits $a \neq b$; $a \oplus b = 0$ if $a = b$. The nth member of the sequence is given by

$$x_n = x_{n-p} \oplus x_{n-q}, \tag{7.58}$$

where $p > q$, and p, q and x_n are integers. The first p random integers must be supplied by another random number generator. As an example of how the operator $\oplus$ works, suppose that $n = 6$, $p = 5$, $q = 3$, $x_3 = 11$, and $x_1 = 6$. Then $x_6 = x_1 \oplus x_3 = 0110 \oplus 1011 = 1101 = 2^3 + 2^2 + 2^0 = 8+4+1 = 13$. Not all values of p and q lead to good results. Some common pairs are $(p, q) = (31, 3), (250, 103)$, and $(521, 168)$. In Java and C the exclusive or operation on the integers `m` and `n` is given by `m^n`. The algorithm for carrying out the above procedure after p integers is produced is shown in the following. Initially the index, k, can be set to 0.

1. If $k < q$, set $j = k + q$, else set $j = k - p + q$.
2. Set $x_k = x_k \oplus x_j$; x_k is the desired random number for this iteration. If a random number between 0 and 1 is desired, divide x_k by the maximum possible integer.
3. Increment k to $(k + 1) \mod p$.

Because the exclusive or operator and bit manipulation is very fast, this random number generator is very fast. However, the period may not be long enough for some applications and the correlations between numbers might not be as good as needed. The shuffling algorithm discussed in Problem 7.36 should be used to improve this generator.

These examples of random number generators illustrate their general nature. That is, numbers in the sequence are used to find the succeeding ones according to a well defined algorithm.[1] The sequence is determined by the *seed*, the first number of the sequence or the first p members of the sequence for the generalized feedback shift register and related generators. Usually, the maximum possible period is related to the size of the computer word, for example, 16, 32, or 64 bits. The choice of the constants and the proper initialization of the sequence is very important and thus these algorithms must be implemented with care.

There is no necessary and sufficient test for the randomness of a finite sequence of numbers; the most that can be said about any finite sequence of numbers is that it is *apparently* random. Because no single statistical test is a reliable indicator, we need to consider several tests. Some of

[1] The quotation by John von Neumann, "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin," is particularly appropriate here.

the best known tests are discussed in Problem 7.35. Note that many of these tests can be stated in terms of random walks.

Problem 7.35. Statistical tests of randomness

a. *Period*. The most obvious requirement for a random number generator is that its period be much greater than the number of random numbers needed in a specific calculation. One way to visualize the period of the random number generator is to use it to generate a plot of the displacement x of a random walker as a function of the number of steps N. When the period of the random number is reached, the plot will begin to repeat itself. Generate such a plot using (7.57) for the two sets of parameters $a = 899$, $c = 0$, and $m = 32768$, and $a = 16807$, $c = 0$, $m = 2^{31} - 1$ with $x_0 = 12$. What are the periods of the corresponding random number generators? Obtain similar plots using different values for the parameters a, c, and m. Why is the seed value $x_0 = 0$ forbidden for te choice $c = 0$? Do some combinations of these parameters give longer periods than others? What is the period of the random number generator that you have been using?

b. *Uniformity*, A random number sequence should contain numbers distributed in the unit interval with equal probability. The simplest test of uniformity is to divide this interval into M equal size subintervals or bins and place each member of the sequence into one of the bins. For example, consider the first $N = 10^4$ numbers generated by (7.57) with $a = 106$, $c = 1283$, and $m = 6075$ (see Press et al.). Place each number into one of $M = 100$ bins. Is the number of entries in each bin approximately equal? What happens if you increase N?

c. *Chi-square test*. Is the distribution of numbers in the bins of part (b) consistent with the laws of statistics? The most common test of this consistency is the *chi-square* or χ^2 test. Let y_i be the observed number in bin i and E_i be the expected value. The chi-square statistic is

$$\chi^2 = \sum_{i=1}^{M} \frac{(y_i - E_i)^2}{E_i}. \tag{7.59}$$

For the example in part (b) with $N = 10^4$ and $M = 100$, we have $E_i = 100$. The magnitude of the number χ^2 is a measure of the agreement between the observed and expected distributions. In general, the individual terms in the sum (7.59) are expected to be order 1, and because there are M terms in the sum, we expect $\chi^2 \leq M$. As an example, we did five independent runs of a random number generator with $N = 10^4$ and $M = 100$, and found $\chi^2 \approx 92$, 124, 85, 91, and 99. These values of χ^2 are consistent with this expectation. Although we usually want χ^2 to be as small as possible, we would be suspicious if $\chi^2 \approx 0$, because such a small value suggests that N is a multiple of the period of the generator and that each value in the sequence appears an equal number of times.

d.* A more quantitative measure of our confidence that the discrepancy $(y_i - E_i)$ is distributed according to a Gaussian distribution is given by the chi-square probability function $P(x, \nu)$ defined as:

$$P(x, \nu) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)} \int_0^x t^{(\nu-2)/2}\, e^{-t/2}\, dt. \tag{7.60}$$

The Gamma function $\Gamma(z)$ in (7.60) is given by $\Gamma(z) = \int_0^\infty t^{z-1}\, e^{-t}\, dt$; the familiar relation $\Gamma(z+1) = z!$ holds if z is a positive integer. The quantity ν in (7.60) is the number of degrees of freedom. In our case $\nu = M-1$, because we have imposed the constraint that $\sum_{i=1}^{M} E_i = N$. The function $Q(x,\nu) = 1 - P(x,\nu)$ is the probability that the measured value of χ^2 is greater than x. For our example we can solve for x in the equation $Q(x,\nu) = q$ with $\nu = 99$ for various values of q or find the solution in a statistical table. (A quick search of the Web will yield several sites that will compute x for particular values of ν and probability q.) For $\nu = 99$, we find that $x \approx 139$ for $q = 0.005$, $x \approx 123$ for $q = 0.05$, $x \approx 111$ for $q = 0.2$, and $x \approx 98$ for $q = 0.5$. Our above results for χ^2 show that $\chi^2 > 123$ for one run out of five (20%). Because 123 is the value of x at the 5% level, we expect to see $\chi^2 \geq 123$ in only one out of twenty runs. Hence, our confidence level is less than 95%. Instead, we can assume an approximately 80% confidence level in our random number generator because the value of x for this confidence level is 111. We might be able to increase our confidence level by doing more runs. Suppose that we make twenty runs and we still find only one measurement of χ^2 greater than 123. In this case our confidence level would rise to 95%. Determine χ^2 for twenty independent runs using the values of a, c, and m given in parts (a) and (b). Estimate your level of confidence in these random number generators.

e. *Filling sites.* Although a random number sequence might contain numbers that are distributed in the unit interval with equal probability, consecutive numbers might not appear in a perfectly uniform way, but have a tendency to be clumped or correlated in some way. One test of this correlation is to fill a square lattice of L^2 sites at random. Consider an array $n(x,y)$ that is initially empty, where $1 \leq x_i, y_i \leq L$. A site is selected randomly by choosing its two coordinates x_i and y_i from two consecutive numbers in the sequence. If the site is empty, it is filled and $n(x_i, y_i) = 1$; otherwise it is not changed. This procedure is repeated t times, where t is the number of Monte Carlo steps per site. Because this process is analogous to the decay of radioactive nuclei, we expect that the fraction of empty lattice sites should decay as e^{-t}. Determine the fraction of unfilled sites using the random number generator that you have been using for $L = 10$, 15, and 20. Are your results consistent with the expected fraction? Repeat the same test using (7.57) with $a = 65549$, $c = 0$, and $m = 231$. The existence of triplet correlations can be determined by a similar test on a simple cubic lattice by choosing the three coordinates x_i, y_i, and z_i from three consecutive random numbers.

f. *Parking lot test.* Fill sites as in part (e) and plot the sites that have been filled. Do the illed sites look random, or are there stripes of filled sites? Try $a = 65549$, $c = 0$, and $m = 231$.

g. *Hidden correlations.* Another way of checking for correlations is to plot x_{i+k} versus x_i. If there are any obvious patterns in the plot, then there is something wrong with the generator. Use the generator (7.57) with $a = 16807, c = 0$, and $m = 2^{31} - 1$. Can you detect any structure in the plotted points for $k = 1$ to $k = 5$? Test the random number generator that you have been using. Do you see any evidence of lattice structure, for example, equidistant parallel lines? Is the logistic map $x_{n+1} = 4x_n(1 - x_n)$ for $r = 1$ a suitable random number generator?

h. *Short-term correlations.* Another measure of short term correlations is the autocorrelation function

$$C(k) = \frac{\langle x_{i+k} x_i \rangle - \langle x_i \rangle^2}{\langle x_i x_i \rangle - \langle x_i \rangle \langle x_i \rangle}, \tag{7.61}$$

where x_i is the ith term in the sequence. We have used the fact that $\langle x_{i+k}\rangle = \langle x_i\rangle$, that is, the choice of the origin of the sequence is irrelevant. The quantity $\langle x_{i+k}x_i\rangle$ is found for a particular choice of k by forming all the possible products of $x_{i+k}x_i$ and dividing by the number of products. If x_{i+k} and x_i are not correlated, then $\langle x_{i+k}x_i\rangle = \langle x_{i+k}\rangle\langle x_i\rangle$ and $C(k) = 0$. Is $C(k)$ identically zero for any finite sequence? Compute $C(k)$ for $a = 106$, $c = 1283$, and $m = 6075$.

i. *Random walk.* A test based on the properties of random walks has been proposed by Vattulainen et al. Assume that a walker begins at the origin of the x-y plane and generate a walk of N steps. Average over n_w trials or walkers and count the number of walks in each quadrant q_i of the x-y plane. Use the χ^2 test (7.59) with $y_i \to q_i$, $M = 4$, and $E_i = n_w/4$. If $\chi^2 > 7.815$ (a 5% probability if the random number generator is perfect), we say that the run fails. The random number generator fails if two out of three independent runs fail. The probability of a perfect generator failing two out of three runs would be approximately $3 \times 0.95 \times (0.05)^2 \approx 0.007$. Test several random number generators.

Problem 7.36. Improving random number generators

One way to reduce sequential correlation and to lengthen the period is to mix or *shuffle* the random numbers produced by a random number generator. The procedure due to Bays and Durham is to begin with a list of N random numbers (between 0 and 1) using a given generator `rng`. The number N is arbitrary, but should be less than the period of `rng`. Also generate one more random number, r_{extra}. Then for each desired random number use the following procedure.

i. Calculate the integer k equal to the truncation of Nr_{extra} to an integer. Use the kth random number, r_k, from your list as the desired random number.

ii. Set r_{extra} equal to the random number, r_k, chosen in step i.

iii. Generate a new random number, r and use it to replace the number chosen in step i, that is, $r_k = r$.

Consider a random number generator with a relatively short period and strong sequential correlation and show that this shuffling scheme improves the quality of the random number sequence.

At least some of the statistical tests given in Problem 7.35 should be done whenever serious calculations are contemplated. However, even if a random number generator passes all these tests, there still can be problems in rare cases. Typically, these problems arise when a small number of events have a large weight. In these cases a very small bias in the random number generator might lead to a systematic error in the final results, and two generators, which appear equally good as determined by various statistical tests, might give statistically different results when used in a specific application. For this reason, it is important that the random number generator that is used be reported along with the actual results. Confidence in the results also can be increased by repeating the calculation with another random number generator.

Because all random number generators are based on a deterministic algorithm, it always is possible to construct a test generator for which a particular algorithm will fail. The success of a random number generator in passing various statistical tests is necessary and improves our overall confidence in its statistical properties, but it is not a sufficient condition for their use in all applications. In Project 15.36 we discuss an application of Monte Carlo methods to the Ising model for which some commonly used random number generators give incorrect results.

7.10 Variational Methods

Many problems in physics can be formulated in terms of a variational principle. In the following, we consider examples of variational principles in geometrical optics and classical mechanics. We then discuss how Monte Carlo methods can be applied to these problems. A more sophisticated application of Monte Carlo methods to a variational problem in quantum mechanics is discussed in Chapter 16.

Our everyday experience of light leads naturally to the concept of light rays. This description of light propagation, called *geometrical* or *ray optics*, is applicable when the wavelength of light is small compared to the linear dimensions of any obstacles or openings. The propagation of light rays can be formulated in terms of Fermat's principle of least time:

> *A ray of light follows the path between two points (consistent with any constraints) that requires the least amount of time.*

Fermat's principle of least time can be adopted as the basis of geometrical optics. For example, Fermat's principle implies that light travels from a point A to a point B in a straight line in a homogeneous medium. Because the speed of light is constant along any path within the medium, the path of shortest time is the path of shortest distance, that is, a straight line from A to B. What happens if we impose the constraint that the light must strike a mirror before reaching B?

The speed of light in a medium can be expressed in terms of c, the speed of light in a vacuum, and the index of refraction n of the medium:

$$v = \frac{c}{n}. \tag{7.62}$$

Suppose that a light ray in a medium with index of refraction n_1 passes through a second medium with index of refraction n_2. The two media are separated by a plane surface. We now show how we can use Fermat's principle and a simple Monte Carlo method to find the path of the light. The analytical solution to this problem using Fermat's principle is found in many texts (cf. Feynman et al.).

Our strategy, as implemented in class `Fermat`, is to begin with a straight path and to make changes in the path at random. These changes are accepted only if they reduce the travel time of the light. Some of the features of `Fermat` and the target class `FermatApp` include:

1. Light propagates from left to right through `N` regions. The index of refraction `n[i]` is uniform in each region `[i]`. The index `i` increases from left to right. We have chosen units such that the speed of light in vacuum equals unity.

2. Because the light propagates in a straight line in each medium, the path of the light is given by the coordinates `y[i]` at each boundary.

3. The coordinates of the light source and the detector are at (0,`y[0]`) and (`N`,`y[N]`) respectively, where `y[0]` and `y[N]` are fixed.

4. The path is the connection of the set of points at the boundary of each region.

5. The path of the light is found by choosing the boundary `i` at random and generating a trial value of `y[i]` that differs from its previous value by a random number between `-dy` to `dy`. If the trial value of `y[i]` yields a shorter travel time, this value becomes the new value for `y[i]`.

6. The path is redrawn whenever it is changed.

Listing 7.6: Fermat class.

```
/*
 * The org.opensourcephysics.sip.ch07 package contains classes for Chapter 7,
 * Random Processes, of the book Simulations in Physics.
 * Copyright (c) 2005 H. Gould, J. Tobochnik, and W. Christian.
 */
package org.opensourcephysics.sip.ch07;

/**
 * Light ray in media with different index of refraction
 *
 * @author Jan Tobochnik
 * modified by H. Gould, 12/8/04, modified by J. Tobochnik, 12/9/04
 */

public class Fermat {

   double y[];   // y coordinate of light ray, index is x coordinate
   double v[];   // light speed of ray for medium starting at index value
   int N;   // number of media
   double dn;  // change in index of refraction from one region to the next
   double dy = 0.1;  // maximum change in y position
   int steps;

   /**
    * Initializes arrays
    */
   public void initialize () {
      y = new double[N+1];
      v = new double[N];
      double indexOfRefraction = 1.0;
      for(int i = 0; i<=N; i++) {
         y[i] = i;   // initial path is a straight line
      }
      for(int i = 0; i<N; i++) {
         v[i] = 1.0/indexOfRefraction;
         indexOfRefraction += dn;
      }
      steps = 0;
   }

   /**
    * Random change in path
```

```
     */
    public void step() {
        int i = 1+(int) (Math.random()*(N-1));
        double yTrial = y[i]+2.0*dy*(Math.random()-0.5);
        double previousTime = Math.sqrt(Math.pow(y[i-1]-y[i], 2)+1)/v[i-1]; // left medium
        previousTime += Math.sqrt(Math.pow(y[i+1]-y[i], 2)+1)/v[i]; // right medium
        double trialTime = Math.sqrt(Math.pow(y[i-1]-yTrial, 2)+1)/v[i-1]; // left medium
        trialTime += Math.sqrt(Math.pow(y[i+1]-yTrial, 2)+1)/v[i]; // right medium
        if(trialTime<previousTime) {
            y[i] = yTrial;
        }
        steps++;
    }
}
```

Listing 7.7: Target class for Fermat's principle.

```
/*
 * The org.opensourcephysics.sip.ch11 package contains classes for Chapter 7,
 * Random Processes, of the book Simulations in Physics.
 * Copyright (c) 2005 H. Gould, J. Tobochnik, and W. Christian.
 */
package org.opensourcephysics.sip.ch07;

import org.opensourcephysics.controls.AbstractSimulation;
import org.opensourcephysics.controls.SimulationControl;
import org.opensourcephysics.frames.PlotFrame;

/**
 * Simulates use of Fermat's principle to find light path that minimizes time
 * @author Jan Tobochnik
 * modified by H. Gould, 12/8/04, modified J. Tobochnik 12/9/04
 */
public class FermatApp extends AbstractSimulation {

    Fermat medium = new Fermat();
    PlotFrame path = new PlotFrame("x", "y", "Light path");

    /**
     * Sets path frame properties
     */
    public FermatApp() {
        path.setAutoscaleX(true);
        path.setAutoscaleY(true);
        path.setConnected(true);   // draw lines between points
    }

    /**
     * Gets parameters and initializes medium
     */
```

```
    public void initialize () {
        medium.dn = control.getDouble("Change in index of refraction");
        medium.N = control.getInt("Number of media segments");
        medium.initialize ();
        path.clearData();
    }

    /**
     * Makes one change in path at a time
     */
    public void doStep() {
        medium.step();
        path.clearData();
        for(int i = 0; i<=medium.N; i++) {
            path.append(0, i, medium.y[i]);
        }
        path.setMessage(medium.steps+" steps");
    }

    /**
     * Resets to default values
     */
    public void reset() {
        control.setValue("Change in index of refraction" , 0.5);
        control.setValue("Number of media segments", 2);
        path.clearData();
    }

    /**
     * Starts the Java application.
     * @param args command line parameters
     */
    public static void main(String[] args) {
        SimulationControl.createApp(new FermatApp());
    }
}
```

Problem 7.37. The law of refraction

a. Use `Fermat` and `FermatApp` to determine the angle of incidence θ_1 and the angle of refraction θ_2 between two media with different indices of refraction. The angles θ_1 and θ_2 are measured from the normal to the boundary. Set $N = 2$ and let the first medium be air ($n_1 \approx 1$) and the second medium be glass ($n_2 \approx 1.5$). Describe the path of the light after a number of trial paths are attempted. Add statements to the program to determine θ_1 and θ_2, the vertical position of the intersection of the light at the boundary between the two media, and the total time for the light to go from (0,`y[0]`) to (2,`y[2]`).

b. Modify the program so that the first medium represents glass ($n_1 \approx 1.5$) and the second medium represents water ($n_2 \approx 1.33$). Verify that your results are consistent with $n_2 \sin\theta_2 = n_1 \sin\theta_1$.

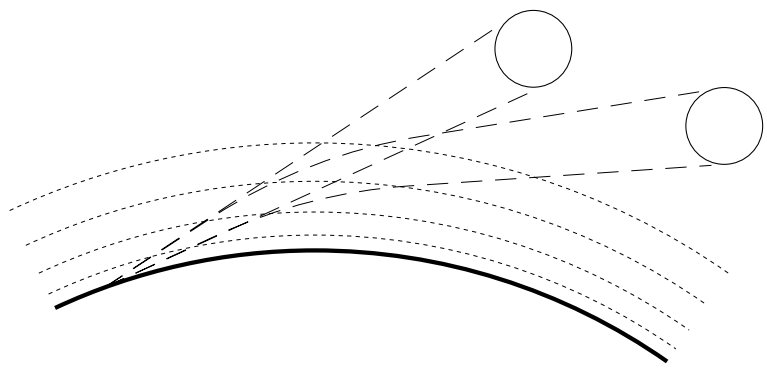

Figure 7.12: Near the horizon, the apparent (exaggerated) position of the sun is higher than the true position of the sun. Note that the light rays from the true sun are curved due to refraction.

Problem 7.38. Inhomogeneous media

a. The earth's atmosphere is thin at the top and dense near the earth's surface. We can model this inhomogeneous medium by dividing the atmosphere into equal width segments each of which is homogeneous. To simulation this atmosphere run your program with `N` $= 10$ and `dn` $= 0.1$, and find the path of least time. Use your results to explain why when we see the sun set, the sun already is below the horizon (see Figure 7.12).

b.* Modify your program to find the appropriate distribution $n(y)$ for a fiber optic cable, which we take to be a flat, long ribbon. In this case the ith region corresponds to a cross sectional slab through the cable. Although a real cable is three-dimensional, we consider a two-dimensional cable for simplicity. We want the cable to have the property that if a ray of light starts from one side of the cable and ends at the other, the slope dy/dx of the path should be near zero at the edges so that light does not escape from the cable.

Fermat's principle is an example of an extremum (maxima or minima) principle. An extremum means that a small change ϵ in an independent variable leads to a change in a function (more precisely, a function of functions) that is proportional to ϵ^2 or a higher power of ϵ. An important extremum principle in classical mechanics is based on the action S:

$$S = \int_{t_0}^{t_{\text{final}}} L\, dt, \tag{7.63}$$

where t_0 and t_{final} are the initial and final times, respectively. The Lagrangian L in (7.63) is the kinetic energy minus the potential energy. The extremum principle for the action is known as *the principle of least action*. If we were to take the extremum of (7.63), we would find that the path for which S is an extremum satisfies Newton's second law (for conservative forces). One reason for the importance of the principle of least action is that quantum mechanics can be formulated in terms of an integral over the action. This way of doing quantum mechanics is called the path integral formulation (see Section 16.10).

To use (7.63) to find the motion of a single particle in one dimension, we fix the position at the chosen initial and final times, $x(t_0)$ and $x(t_{\text{final}})$, and then choose the velocities and positions for the intermediate times $t_0 < t < t_{\text{final}}$ to minimize the action. One way to implement this procedure numerically is to convert the integral in (7.63) to a sum:

$$S \approx \sum_{i=1}^{N-1} L(t_i)\,\Delta t, \tag{7.64}$$

where $t_i = t_0 + i\Delta t$. (The approximation used to obtain (7.64) is known as the rectangular approximation and is discussed in Chapter 11.) For a single particle in one dimension, we can write

$$L_i \approx \frac{m}{2(\Delta t)^2}(x_{i+1} - x_i)^2 - u(x_i), \tag{7.65}$$

where m is the mass of the particle and $u(x_i)$ is the potential energy of the particle at x_i. The velocity has been approximated as the difference in position divided by the change in time Δt.

Problem 7.39. Principle of least action

a. Write a program to minimize the action S given in (7.63) for the motion of a single particle in one dimension. Use the approximate form of the Lagrangian given in (7.65). One way to write the program is to modify class `Fermat` so that the vertical coordinate for the light ray becomes the position of the particle, and the horizontal region number `i` becomes the discrete time interval of duration Δt. The quantity to be minimized is different, but otherwise the algorithm is similar.

b. Verify your program for the case of free fall for which the potential energy is $u(x) = mgx$. Choose $x(t = 0) = 2\,\text{m}$ and $x(t = 10\,\text{s}) = 8\,\text{m}$, and begin with $N = 20$. Allow the maximum change in the position to be $5\,\text{m}$.

c. Consider the harmonic potential $u(x) = \frac{1}{2}kx^2$. What shape do you expect the path $x(t)$ to be? Increase N to approximately 50 and estimate the path by minimizing the action.

It is possible to extend the principle of least action to more dimensions or particles, but it is necessary to begin with a path close to the optimum one to obtain a good approximation to the optimum path in a reasonable time.

In Problems 7.37–7.39 a simple Monte Carlo algorithm that always accepts paths ehich reduce the time or action is sufficient. However, for more complicated index of refraction distributions or potentials, it is possible that such a simple algorithm will find only a local minimum and the global minimum will be missed. The problem of finding the global minimum is very general and is shared by all optimization algorithms if the system has many relative minima. Optimization is a very active area of research in many fields of science and engineeering. Ideas from physics, biology, and computer science have led to many improved algorithms. We will discuss some of these algorithms in Chapter 15. In most of these algorithms paths that are worse than the current path are sometimes accepted in an attempt to climb out of a local minimum. Other algorithms involve ways of sampling over a wider range of possible paths. Another approach is to convert

the Monte Carlo algorithm into a deterministic algorithm. We have already mentioned that an analytical variational calculation leads to Newton's second law. Passerone and Parrinello discuss an algorithm for looking for extrema in the action by maintaining the discrete structure in Eq. (7.65), and then finding the extremum by taking the derivative with respect to each coordinate, x_i and setting the resulting equations equal to zero. This procedure leads to a set of derministic equations that need to be solved numerically. By enforcing energy conservation and adding a few other tricks to improve performance, they obtain a more efficient algorithm.

7.11 Projects

Almost all of the problems in this chapter can be done using more efficient programs, greater number of trials, and larger systems. More applications of random walks and random number sequences are discussed in subsequent chapters. Many more ideas for projects can be gained from the references.

Project 7.40. Competition between diffusion and fragmentation

As we have discussed, random walks are useful for understanding diffusion in contexts more general than the moves of a particle. For example, consider a particle in solution that can either grow by the absorption of particles or shrink by the random loss of small particles. We can model this process as a random walk where the spatial coordinate is replaced by the mass of the particle. One difference between this situation and a standard random walk is that the random variable, the mass, must be positive. There are other mechanisms that can change the size of the particle. For example, the particle could split into two fragments. Recently, a model of diffusion plus fragmentation was explored analytically, and it was found that the size distribution of the particles had a universal form (see Ferkinghoff-Berg et al.). The results of the model were compared with experimental data on ice crystal sizes and the length distribution of α helices in proteins. We explore this behavior in the following.

a. Write a program to simulate diffusion plus fragmentation in one dimension. Begin with N objects with some distribution of lengths. Let the integer L_i represent the length of the ith object. At each step all the objects change their length by ± 1. If the length of an object is equal to 0, it is removed from the system. An easy way to eliminate the ith object is to set its length equal to the length of the last object and reduce N by unity. Next, at each step choose only one object at random to fragment with a probability that is proportional to the length of the object. One way to do so in given in the following code, where `totalMass` is the sum of the lengths of all the objects.

```
// choose random location in object where fragmentation occurs
int x = (int)(Math.random()*totalMass);
int i = 0;   // label for object
int sum = length[i];
while(sum < x) { // find object to fragment
   i++;
   sum += length[i];
}
if(length[i] > 1) {
```

```
    int pieceA = 1 + (int)(Math.random()*(length[i]-1));
    int pieceB = length[i] - pieceA;
    length[i] = pieceA;
    length[numberOfObjects] = pieceB; // new object
    numberOfObjects++;
  }
```

b. The key quantity of interest is the distribution of lengths $P(L)$, which is proportional to the mean number of objects of length L. Include this calculation in your program and plot the distribution periodically. Do not include the initial transient behavior in the averages. Explore a variety of initial length distributions with about $N = 5000$ objects. Is the long time distribution similar in shape for any initial distribution? Calculate the total mass (sum of the lengths) and output this value periodically. Although this quantity will fluctuate, it should remain approximately constant. Why?

c. Collect data for three different initial distributions and scale $P(L)$ and L so that the three distributions roughly fall onto the same curve. For example, you can scale $P(L)$ so that all three distributions have the same maximum height. Then try multiplying each L by a factor so that the distributions overlap.

d. The analytical results suggest that the universal behavior can be obtained by scaling L by the total mass m raised to the 1/3 power. Is this analytical prediction consistent with your results? Test this hypothesis by adjusting the initial distributions so that they all have the same total mass. Your results for the long time length distribution should fall on a universal curve. Why is this universality interesting? How can this result be used to analyze different systems? Does one need to do a new simulation for each different system?

e. What happens if the fragmentation step is done less or more often than every random walk step. Does the scaling change?

Project 7.41. Application of the pivot algorithm to self-avoiding walks

The algorithms that we have discussed for generating self-avoiding random walks are all based on making *local* deformations of the walk (polymer chain) for a given value of N, the number of bonds. As discussed in Problem 7.31, the time τ between statistically independent configurations is nonzero. The problem is that τ increases with N as some power, for example, $\tau \sim N^3$. This power law dependence of τ on N is called *critical slowing down* and implies that it becomes increasingly more time consuming to generate long walks. We now discuss an example of a *global* algorithm that reduces the dependence of τ on N. Another example of a global algorithm that reduces critical slowing down is discussed in Project 15.34.

a. Consider the walk shown in Figure 7.13a. Select a site at random and one of the four possible directions. The shorter portion of the walk is rotated (pivoted) to this new direction by treating the walk as a rigid structure. The new walk is accepted only if the new walk is self-avoiding; otherwise the old walk is retained. (The shorter portion of the walk is chosen to save computer time.) Some typical moves are shown in Figure 7.13. Note that if an end point is chosen, the previous walk is retained. Write a program to implement this algorithm and compute the dependence of the mean square end-to-end distance $\langle R_N^2 \rangle$ on N. Consider values of N in the

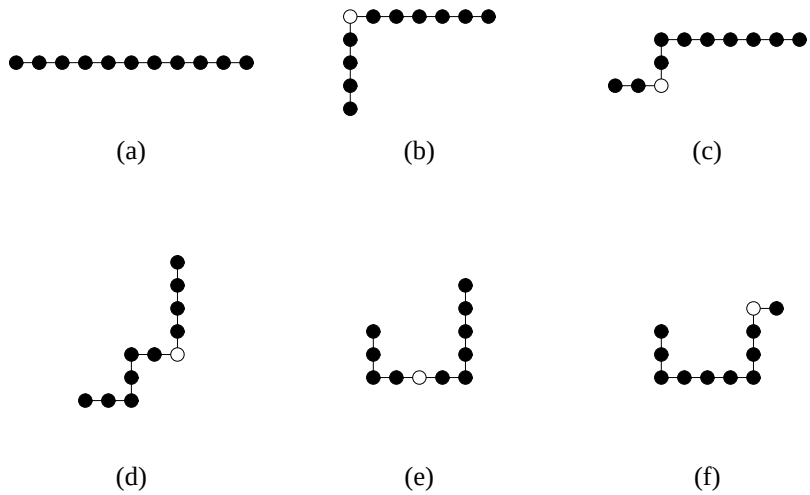

Figure 7.13: Examples of the first several changes generated by the pivot algorithm for a self-avoiding walk of $N = 10$ steps (11 sites). The open circle denotes the pivot point. This figure is adopted from the article by MacDonald et al.

range $10 \leq N \leq 80$. A discussion of the results and the implementation of the algorithm can be found in MacDonald et al. and Madras and Sokal, respectively.

b. Compute the correlation time τ for different values of N using the approach discussed in Problem 7.31b.

Project 7.42. Pattern formation

In Problem 7.34 we saw that simple patterns can develop as a result of random behavior. The phenomenon of pattern formation is of much interest in a variety of contexts ranging from the large scale structure of the universe to the roll patterns seen in convection (for example, smoke rings). In the following, we explore the patterns that can develop in a simple reaction diffusion model based on the reactions, $A + 2B \rightarrow 3B$, and $B \rightarrow C$, where C is inert. Such a reaction is called *autocatalytic*.

In Problem 7.34 we considered chemical reactions in a closed system where the reactions can proceed to equilibrium. In contrast, open systems allow a continuous supply of fresh reactants and a removal of products. These two processes allow steady states to be realized and oscillatory conditions to be maintained indefinitely. In this problem we assume that A is added at a constant rate and that both A and B are removed by the feed process. Pearson (see references) modeled these processes by two coupled reaction diffusion equations:

$$\frac{\partial A}{\partial t} = D_A \nabla^2 A - AB^2 + f(1 - A) \tag{7.66a}$$

$$\frac{\partial B}{\partial t} = D_B \nabla^2 B + AB^2 - (f + k)B. \tag{7.66b}$$

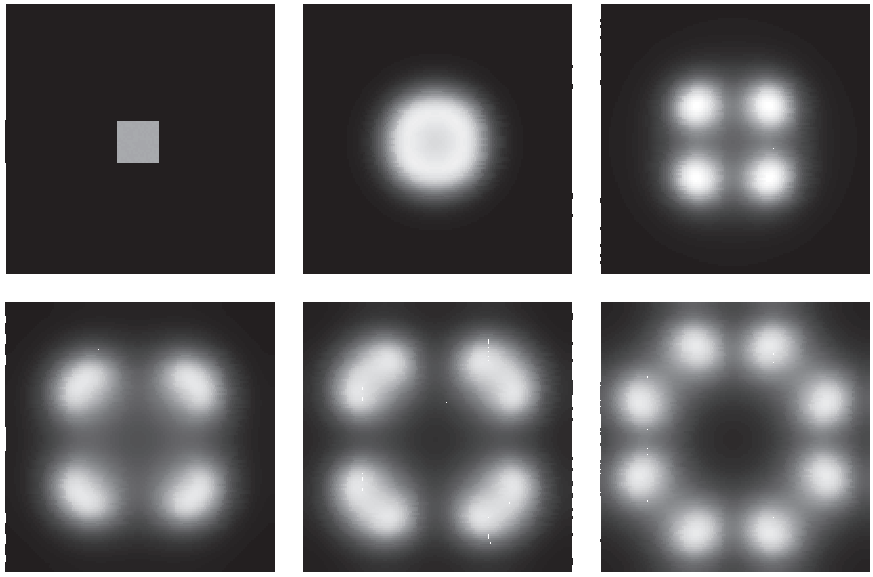

Figure 7.14: Evolution of the pattern starting from the initial conditions suggested in Project c.

The AB^2 term represents the reaction $A + 2B \to 3B$. This term is negative in (7.66a) because the reactant A decreases, and is positive in (7.66b) because the reactant B increases. The term $+f$ represents the constant addition of A, and the terms $-fA$ and $-fB$ represent the removal process; the term $-kB$ represents the reaction $B \to C$. All the quantities in (7.66) are dimensionless. We assume that the diffusion coefficients are $D_A = 2 \times 10^{-5}$ and $D_B = 10^{-5}$, and the behavior of the system is determined by the values of the rate constant k and the feed rate f.

a. We first consider the behavior of the reaction kinetics that results when the diffusion terms in (7.66) are neglected. It is clear from (7.66) that there is a trivial steady state solution $A = 1$, $B = 0$. Are there other solutions, and if so, are they stable? The steady state solutions can easily be found by solving (7.66) with $\partial A/\partial t = \partial B/\partial t = 0$. To determine the stability, we can add a perturbation and determine whether the perturbation grows or not. However, without the diffusion terms, it is more straightforward to solve (7.66) numerically using a simple Euler algorithm. Choose a time step equal to unity, and let $A = 0.1$ and $B = 0.5$ at $t = 0$. Determine the steady state values for $0 < f \leq 0.3$ and $0 < k \leq 0.07$ in increments of $\Delta f = 0.02$ and $\Delta k = 0.005$. Record the steady state values of A and B. Then repeat this exercise for the initial values $A = 0.5$ and $B = 0.1$. You should find that for some values of f and k, only one steady state solution can be obtained for the two initial conditions, and for other initial values of A and B there are two steady state solutions. Try other initial conditions. If you obtain a new solution, change the initial A or B slightly to see if your new solution is stable. On an f versus k plot indicate where there are two solutions and where there are one. In this way you can determine the approximate phase diagram for this process.

b. There is a small region in f-k space where one of the steady state solutions becomes unstable

and periodic solutions occur (the mechanism is known as a Hopf bifurcation). Try $f = 0.009$, $k = 0.03$, and set $A = 0.1$ and $B = 0.5$ at $t = 0$. Plot the values of A and B versus the time t. Are they periodic? Try other values of f and k and estimate where the periodic solutions occur.

c. Numerical solutions of the full equation with diffusion (7.66) can be found by making a finite difference approximation to the spatial derivatives as in (3.15) and using a simple Euler algorithm for the time integration. Adopt periodic boundary conditions. Although it is straightforward to write a program to do the numerical integration, an exploration of the dynamics of this system requires much computer resources. However, we can find some preliminary results with a small system and a coarse grid. Consider a 0.5×0.5 system with a spatial mesh of 128×128 grid points on a square lattice. Choose $f = 0.18$, $k = 0.057$, and $\Delta t = 0.1$. Let the entire system be in the initial trivial state ($A = 1$, $B = 0$) except for a 20×20 grid located at the center of the system where the sites are $A = 1/2$, $B = 1/4$ with a $\pm 1\%$ random noise. The effect of the noise is to break the square symmetry. Let the system evolve for approximately 80,000 time steps and look at the patterns that develop. Color code the grid according to the concentration of A, with red representing $A = 1$ and blue representing $A \approx 0.2$ and with several intermediate colors. Very interesting patterns have been found by Pearson.

Appendix 7: Random Walks and the Diffusion Equation

To gain some insight into the relation between random walks and the diffusion equation, we first show that the latter implies that $\langle x(t)\rangle$ is zero and $\langle x^2(t)\rangle$ is proportional to t. We rewrite the diffusion equation (7.25) here for convenience:

$$\frac{\partial P(x,t)}{\partial t} = D\frac{\partial^2 P(x,t)}{\partial x^2}. \tag{7.67}$$

To derive the t dependence of $\langle x(t)\rangle$ and $\langle x^2(t)\rangle$ from (7.67), we write the average of any function of x as

$$\langle f(x,t)\rangle = \int_{-\infty}^{\infty} f(x)P(x,t)\,dx. \tag{7.68}$$

The average displacement is given by

$$\langle x(t)\rangle = \int_{-\infty}^{\infty} xP(x,t)\,dx. \tag{7.69}$$

To do the integral on the right hand side of (7.69), we multiply both sides of (7.67) by x and formally integrate over x:

$$\int_{-\infty}^{\infty} x\frac{\partial P(x,t)}{\partial t}\,dx = D\int_{-\infty}^{\infty} x\frac{\partial^2 P(x,t)}{\partial x^2}\,dx. \tag{7.70}$$

The left-hand side can be expressed as:

$$\int_{-\infty}^{\infty} x\frac{\partial P(x,t)}{\partial t}\,dx = \frac{\partial}{\partial t}\int_{-\infty}^{\infty} xP(x,t)\,dx = \frac{d}{dt}\langle x\rangle. \tag{7.71}$$

The right-hand side of (7.70) can be written in the desired form by doing an integration by parts:

$$D\int_{-\infty}^{\infty} x\frac{\partial^2 P(x,t)}{\partial x^2}\,dx = D\,x\frac{\partial P(x,t)}{\partial x}\bigg|_{x=-\infty}^{x=\infty} - D\int_{-\infty}^{\infty}\frac{\partial P(x,t)}{\partial x}\,dx. \tag{7.72}$$

The first term on the right hand side of (7.72) is zero because $P(x=\pm\infty,t)=0$ and all the spatial derivatives of P at $x=\pm\infty$ are zero. The second term also is zero because it integrates to $D[P(x=\infty,t)-P(x=-\infty,t)]$. Hence, we find that

$$\frac{d\langle x\rangle}{dt} = 0, \tag{7.73}$$

or $\langle x\rangle$ is a constant, independent of time. Because $x=0$ at $t=0$, we conclude that $\langle x\rangle = 0$ for all t.

To calculate $\langle x^2(t)\rangle$, two integrations by parts are necessary, and we find that

$$\frac{d}{dt}\langle x^2(t)\rangle = 2D, \tag{7.74}$$

or

$$\langle x^2(t)\rangle = 2Dt. \tag{7.75}$$

We see that the random walk and the diffusion equation have the same time dependence. In d-dimensional space, $2D$ is replaced by $2dD$.

The solution of the diffusion equation shows that the time dependence of $\langle x^2(t)\rangle$ is equivalent to the long time behavior of a simple random walk on a lattice. In the following, we show directly that the continuum limit of the one-dimensional random walk model is a diffusion equation.

If there is an equal probability of taking a step to the right or left, the random walk can be written in terms of the simple master equation

$$P(i,N) = \frac{1}{2}[P(i+1,N-1)+P(i-1,N-1)], \tag{7.76}$$

where $P(i,N)$ is the probability that the walker is at site i after N steps. To obtain a differential equation for the probability density $P(x,t)$, we identify $t=N\tau$, $x=ia$, and $P(i,N)=aP(x,t)$, where τ is the time between steps and a is the lattice spacing. This association allows us to rewrite (7.76) in the equivalent form

$$P(x,t) = \frac{1}{2}[P(x+a,t-\tau)+P(x-a,t-\tau)]. \tag{7.77}$$

We rewrite (7.77) by subtracting $P(x,t-\tau)$ from both sides of (7.77) and dividing by τ:

$$\frac{1}{\tau}\big[P(x,t)-P(x,t-\tau)\big] = \frac{a^2}{2\tau}\big[P(x+a,t-\tau)-2P(x,t-\tau)+P(x-a,t-\tau)\big]a^{-2}. \tag{7.78}$$

If we expand $P(x,t-\tau)$ and $P(x\pm a,t-\tau)$ in a Taylor series and take the limit $a\to 0$ and $\tau\to 0$ with the ratio $D\equiv a^2/2\tau$ finite, we obtain the diffusion equation

$$\frac{\partial P(x,t)}{\partial t} = D\,\frac{\partial^2 P(x,t)}{\partial x^2}. \tag{7.79a}$$

The generalization of (7.79a) to three dimensions is

$$\frac{\partial P(x,y,z,t)}{\partial t} = D\,\nabla^2 P(x,y,z,t), \tag{7.79b}$$

where $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2 + \partial^2/\partial x^2$ is the Laplacian operator. Equation (7.79) is known as the *diffusion* equation and is frequently used to describe the dynamics of fluid molecules.

The direct numerical solution of the prototypical *parabolic* partial differential equation (7.79) is a nontrivial problem in numerical analysis (cf. Press et al. or Koonin and Meredith.) An indirect method of solving (7.79) numerically is to use a Monte Carlo method, that is, replace the partial differential equation (7.79) by a corresponding random walk on a lattice with discrete time steps. Because the asymptotic behavior of the partial differential equation and the random walk model are equivalent, this approach uses the Monte Carlo technique as a method of *numerical analysis.* In contrast, if our goal is to understand a random walk lattice model directly, the Monte Carlo technique is a *simulation* method. The difference between simulation and numerical analysis is sometimes in the eyes of the beholder.

Problem 7.43. Biased random walk

Show that the form of the differential equation satisfied by $P(x,t)$ corresponding to a random walk with a drift, that is, a walk for $p \neq q$, is

$$\frac{\partial P(x,t)}{\partial t} = D\,\nabla^2 P(x,y,z,t) - v\frac{\partial P(x,t)}{\partial x}. \tag{7.80}$$

How is v related to p and q?

References and Suggestions for Further Reading

Daniel J. Amit, G. Parisi, and L. Peleti, "Asymptotic behavior of the "true" self-avoiding walk," Phys. Rev. B **27**, 1635–1645 (1983).

Panos Argyrakis, "Simulation of diffusion-controlled chemical reactions," Computers in Physics **6**, 525 (1992).

Carter Bays and S. D. Durham, "Improving a poor random number generator," ACM Trans. Math. Software **2**, 59 (1976). Uses a shuffling algorithm to improve random number generators.

William R. Bennett, *Scientific and Engineering Problem-Solving with the Computer*, Prentice Hall (1976). Many random processes including the spread of disease are considered in Chapter 6.

J. M. Bernardo and A. F. M. Smith, *Bayesian Theory*, John Wiley & Sons (1994). Bayes theorem is given concisely on page 2.

J. Bernasconi and L. Pietronero, "True self-avoiding walk in one dimension," Phys. Rev. B **29**, 5196 (1984). The authors present results for the exponent ν accurate to 1%.

Philip R. Bevington and D. Keith Robinson, *Data Reduction and Error Analysis for the Physical Sciences*, second edition, McGraw-Hill (1992).

I. Carmesin and Kurt Kremer, "The bond fluctuation model: A new effective algorithm for the dynamics of polymers in all spatial dimensions," Macromolecules **21**, 2819–2823 (1988).

S. Chandrasekhar, "Stochastic problems in physics and astronomy," Rev. Mod. Phys. **15**, 1 (1943). This article is reprinted in M. Wax, *Selected Papers on Noise and Stochastic Processes*, Dover (1954).

William S. Cleveland and Robert McGill, "Graphical perception and graphical methods for analyzing scientific data," Science **229**, 828 (1985). There is more to analyzing data than least squares fits.

Mohamed Daoud, "Polymers," Chapter 6 in Armin Bunde and Shlomo Havlin, editors, *Fractals in Science*, Springer-Verlag (1994).

Roan Dawkins and Daniel ben-Avraham, "Computer simulations of diffusion-limited reactions," Comput. Sci. Eng. **3**(1), 72–76 (2001).

A. K. Dewdney, "Computer Recreations (Five easy pieces for a do loop and random-number generator)," Sci. Amer. **252** (4), 20–30 (1985).

Robert M. Eisberg, *Applied Mathematical Physics with Programmable Pocket Calculators*, McGraw-Hill (1976). Chapter 7 discusses entropy and the arrow of time.

R. Everaers, I. S. Graham, and M. J. Zuckermann, "End-to-end distance and asymptotic behavior of self-avoiding walks in two and three dimensions," J. Phys. A **28**, 1271 (1995).

Jesper Ferkinghoff-Borg, Mogens H. Jensen, Joachim Mathiesen, Poul Olesen, and Kim Sneppen, "Competition between diffusion and fragmentation: An important rvolutionary process of nature," Phys. Rev. Lett. **91**, 266103-1–4 (2003).

Richard P. Feynman, Robert B. Leighton, and Matthew Sands, The Feynman Lectures on Physics, Addison-Wesley (1963). See Vol. 1, Chapter 26 for a discussion of the principle of least time and Vol. 2, Chapter 19, for a discussion of the principle of least action.

Pierre-Giles de Gennes, *Scaling Concepts in Polymer Physics,* Cornell University Press (1979). An important but difficult text.

Peter Grassberger, "Pruned-enriched Rosenbluth method: Simulations of u polymers of chain length up to 1 000 000," Phys. Rev. E **56**, 3682 (1997).

Shlomo Havlin and Daniel Ben-Avraham, "Diffusion in disordered media," Adv. Phys. **36**, 695 (1987). Section 7 of this review article discusses trapping and diffusion-limited reactions. See also Daniel Ben-Avraham and Shlomo Havlin, *Diffusion and Reactions in Fractals and Disordered Systems*, Cambridge University Press (2001).

Shlomo Havlin, George H. Weiss, James E. Kiefer, and Menachem Dishon, "Exact enumeration of random walks with traps," J. Phys. A: Math. Gen. **17**, L347, (1984). The authors discuss a method based on exact enumeration for calculating the survival probability of random walkers on a lattice with randomly distributed traps.

Brian Hayes, "How to Avoid Yourself," Amer. Scientist **86** (4), 314 (1998).

Z. Jiang and C. Ebner, "Simulation study of reaction fronts," Phys. Rev. A **42**, 7483 (1990).

Peter R. Keller and Mary M. Keller, *Visual Cues*, IEEE Press (1993). A well illustrated book on data visualization techniques.

Steven E. Koonin and Dawn C. Meredith, *Computational Physics*, Addison-Wesley (1990).

Donald E. Knuth, *Seminumerical Algorithms*, second edition, Vol. 2 of *The Art of Computer Programming,* Addison-Wesley (1981). The standard reference on random number generators.

Bruce MacDonald, Naeem Jan, D. L. Hunter, and M. O. Steinitz, "Polymer conformations through 'wiggling'," J. Phys. A **18**, 2627 (1985). A discussion of the pivot algorithm discussed in Project 7.41.

Vishal Mehra and Peter Grassberger, "Trapping reaction with mobile traps," Phys. Rev. E **65**, 050101(R) (2002). This paper discusses the model of a single walker moving on a lattice of traps.

Elliott W. Montroll and Michael F. Shlesinger, "On the wonderful world of random walks," in *Nonequilibrium Phenomena II: From Stochastics to Hydrodynamics,* J. L. Lebowitz and E. W. Montroll, editors, North-Holland Press (1984). The first part of this delightful review article chronicles the history of the random walk.

M. E. J. Newman and G. T. Barkema, *Monte Carlo Methods in Statistical Physics*, Oxford University (1999). This book has a good section on random number generators.

Daniele Passerone and Michele Parrinello, "Action-derived molecular dynamics in the study of rare events," cond-mat/0104204. This paper describes a determininstic algorithm for finding extrema of the action.

John E. Pearson, "Complex patterns in a simple fluid," Science **261**, 189 (1993). See also P. Gray and S. K. Scott, "Sustained oscillations and other exotic patterns of behavior in isothermal reactions," J. Phys. Chem. **89**, 22 (1985).

Thomas Prellberg, T. (2001). "Scaling of self-avoiding walks and self-avoiding trails in three dimensions," Journal of Physics A **34**, 599 (2001). The author estimates that $\nu \approx 0.5874(2)$ for the self-avoiding walk in three dimensions.

Thomas Prellberg and Jaroslaw Krawczyk, "Flat histogram version of the pruned and enriched Rosenbluth method," Phys. Rev. Lett. **92**, 120602 (2004). The authors discusses an improved algorithm for simulating self avoiding walks.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes*, second edition, Cambridge University Press (1992). See Chapter 15 for a general discussion of the modeling of data including general linear least squares and nonlinear fits.

Sidney Redner and Francois Leyvraz, "Kinetics and spatial organization of competitive reactions," Chapter 7 in Armin Bunde and Shlomo Havlin, editors, *Fractals in Science*, Springer-Verlag (1994).

F. Reif, *Fundamentals of Statistical and Thermal Physics*, McGraw-Hill (1965). This well known text on statistical physics has a good discussion on random walks (Chapter 1) and diffusion (Chapter 12).

F. Reif, *Statistical and Thermal Physics*, Berkeley Physics, Vol. 5, McGraw-Hill (1965). Chapter 2 introduces random walks.

Marshall N. Rosenbluth and Arianna W. Rosenbluth, "Monte Carlo calculation of the average extension of molecular chains," J. Chem. Phys. **23**, 356 (1955). One of the first Monte Carlo calculations for the self-avoiding walk.

David Ruelle, *Chance and Chaos*, Princeton Publishing Company (1993). A non-technical introduction to chaos theory that discusses the relation of chaos to randomness.

Charles Ruhla, *The Physics of Chance*, Oxford University Press (1992). A delightful book on probability in many contexts.

Andreas Ruttor, Georg Reents, and Wolfgang Kinzel, "Synchronization of random walks with reflecting boundaries," J. Phys. A: Math. Gen. **37**, 8609–8618 (2004).

G. L. Squires, *Practical Physics*, third edition, Cambridge University Press (1985). An excellent text on the design of experiments and the analysis of data.

John R. Taylor, *An Introduction to Error Analysis*, University Science Books, Oxford University Press (1982).

Edward R. Tuffe, *The Visual Display of Quantitative Information*, Graphics Press (1983).

I. Vattulainen, T. Ala-Nissila, and K. Kankaala, "Physical tests for random numbers in simulations," Phys. Rev. Lett. **73**, 2513 (1994).

Peter H. Verdier and W. H. Stockmayer, "Monte Carlo calculations on the dynamics of polymers in dilute solution," J. Chem. Phys. **36**, 227 (1962).

Frederick T. Wall and Frederic Mandel, "Macromolecular dimensions obtained by an efficient Monte Carlo method without sample attrition," J. Chem. Phys. **63**, 4592 (1975). An exposition of the reptation method.

George H. Weiss, "A primer of random walkology," Chapter 5 in Armin Bunde and Shlomo Havlin, editors, *Fractals in Science*, Springer-Verlag (1994).

George H. Weiss and Shlomo Havlin, "Trapping of random walks on the line," J. Stat. Phys. **37**, 17 (1984). The authors discuss an analytical approach to the asymptotic behavior of one-dimensional random walkers with randomly placed traps.

George H. Weiss and Robert J. Rubin, "Random walks: theory and selected applications," Adv. Chem Phys. **52**, 363 (1983). In spite of its research orientation, much of this review article can be understood by the well motivated student.

Charles A. Whitney, *Random Processes in Physical Systems*, John Wiley and Sons (1990). An excellent introduction to random processes with many applications to astronomy.

Robert S. Wolff and Larry Yaeger, *Visualization of Natural Phenomena*, Springer-Verlag (1993). A CD-ROM disk is included with the book. An extensive list of references also is given.

Hugh D. Young, *Statistical Treatment of Experimental Data*, McGraw-Hill (1962).