

Chapter 7

Random Processes

©2001 by Harvey Gould, Jan Tobochnik, and Wolfgang Christian
4 April 2001

Random processes are introduced in the context of several simple physical systems.

7.1 Order to Disorder

In Chapter 6 we saw several examples of how the under certain conditions, the behavior of nonlinear deterministic systems can be so complicated that it can be described as 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 enough on the outcome of a game for which the outcome is determined by chance, we will lose money eventually if the probability of winning is less than 50%.

We first give an example that illustrates the tendency of many particle systems 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 system reaches equilibrium, and the average number of particles in each half of the box is $N/2$.

How can we simulate this process? One way is to give each particle an initial velocity and position and adopt a simple deterministic model of the motion of the particles. 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 consider a simpler approach based on the simulation of a *random process*.

The basic assumption underlying the probabilistic model that we will consider is that the motion of the particles is such that their trajectory 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 is the same for all particles regardless of the number of particles in either half.

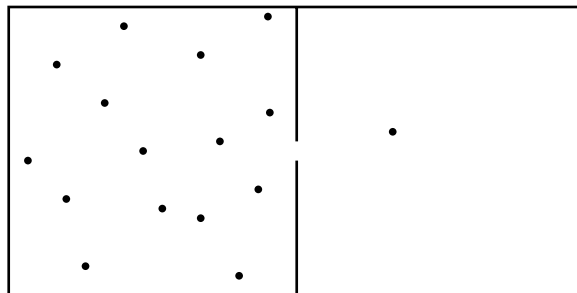


Figure 7.1: A box is divided into two equal halves by a partition. After a small hole is opened in the partition, one particle can pass through the hole per unit time.

We also assume that the size of the hole is such that only one particle can pass through it in one unit of time.

Assume that our model consists of N noninteracting particles, all of which are initially on the left-hand side. One way to implement move a particle is to choose a particle at random and move it to the other side. For example, we could use an array to specify the position of each particle and simulate the random process by generating at random an integer i between 1 and N and changing the array appropriately. The tool we need to do this simulation and other simulations of probabilistic systems is a random number generator.

It might seem strange that we can use a deterministic computer to generate sequences of random numbers. In Chapter 12 we discuss some of the methods for computing a set of numbers that 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 various programming languages, although these random number generators 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 1 and N we can write:

```
int i = (int)(N*Math.random()) + 1;
```

Because the effect of the `int` function is to round the output of `rnd` to its nearest integer, it is necessary to add 1.

The procedure we have specified is needlessly cumbersome, because our only interest is the number of particles on each side. That is, we need to know only n , the number of particles on the left side; the number on the right side is $n' = N - n$. 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 side divided by the total number of particles, that is, the probability of a move from left to right is n/N . The algorithm for simulating the evolution of the model can be summarized by the following steps:

1. Generate a random number r from a uniformly distributed set of random numbers in the interval $0 \leq r < 1$.
2. Compare r to the current value of the fraction of particles n/N on the left side of the box.
3. If $r \leq n/N$, move a particle from left to right, that is, let $n \rightarrow n + 1$; otherwise, move a particle from right to left.
4. Increase the “time” by unity.

Note that the above definition of time is arbitrary. Class `Box` implements this algorithm and plots the evolution of n .

```
// 4/5/01, 10:20 am
package edu.clarku.sip.chapter7;
import edu.clarku.sip.plot.*;
import edu.clarku.sip.templates.*;

// simulation of the particles in a box problem
public class Box implements AnimationModel, Runnable
{
    // N should be lower case according to Java convention for variables
    private int N;          // total number of particles
    private int nleft;     // number on left
    private int time;
    private Control myControl = new SAnimationControl(this);
    private Plot plot;
    private Thread thread;
    private int dataSet = -1;

    public Box()
    {
        plot = new Plot("time","nleft", "Approach to equilibrium");
        plot.setYMinimum(0);
    }

    public void step()
    {
        plot.addPoint(dataSet,time,nleft);
        plot.render();
        moveParticle();
    }

    public void reset()
    {
        myControl.setValue("N", 64);
    }
}
```

```
public void stopCalculation()
{
    thread = null;
}

public void continueCalculation()
{
    thread = new Thread(this);
    thread.start();
}

public void clear(){plot.deleteAllPoints(); dataSet = -1;}

public void startCalculation()
{
    dataSet++;
    N = (int) myControl.getValue("N");
    time = 0;
    nleft = N;          // all particles initially on left side
    thread = new Thread(this);
    thread.start();
}

public void run()
{
    while (thread == Thread.currentThread())
        step();
}

// move particle through hole
public void moveParticle()
{
    // generate random number and move particle
    double r = Math.random();
    double ratio = (double) nleft/N;
    if (r <= ratio)
        nleft--;
    else
        nleft++;
    time++;
}

public static void main(String[] args)
{
    Box b = new Box();
```

```

        b.reset();
    }
}

```

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

- It is frequently quicker to write a short program to test the properties of a function than to look it up in a manual. For example, what are the values of `(int) 3/2` and `(int) -3/2`.
- Write a little program to test the nature of `Math.round(arg)`, `Math.ceil(arg)`, and `Math rint`.
- Write a short program to test whether the same sequence of random numbers appears each time the program is run if we use the method class `Math.random()` to generate the sequence. In Chapter 12 we will learn about ways of generating random numbers and ways that we can set the seed in Java so that we can generate the same sequence if we wish. The ability to do the latter is essential for testing your programs.

Problem 7.2. Approach to equilibrium

- Run the program and describe the time 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 . Estimate the time for the system to reach equilibrium from the plots. What is your qualitative criterion for equilibrium. How does this time depend on N ? What criterion did you use for equilibrium? Does n change when the system is in equilibrium?
- 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?

Problem 7.3. Equilibrium fluctuations

- Modify the program so that averages are taken after equilibrium has been reached. What is the maximum deviation of $n(t)$ from $N/2$ for $N = 64, 400, 800$, and 3600 ? Run for a time that is long enough to yield meaningful results. How do you know if your results are meaningful? How do your results for the maximum deviation depend on N ?
- A measure of the equilibrium fluctuations is the variance σ^2 defined as

$$\sigma^2 = \overline{(n - \bar{n})^2} = \overline{n^2} - \bar{n}^2. \quad (7.1)$$

The bar denotes a time average taken after the system has reached equilibrium. The relative magnitude of the fluctuations is σ/\bar{n} . Compute the variance of n for the same values of N considered in part (a). How do the relative fluctuations, σ/\bar{n} , depend on N ?

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 $N \gg 1$. 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 $n(t)$ in the time interval Δt is given by

$$\Delta n = \left[\frac{-n(t)}{N} + \frac{N - n(t)}{N} \right] \Delta t. \quad (7.2)$$

(Recall that 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 n and t as continuous variables and take the limit $\Delta t \rightarrow 0$, we have

$$\frac{dn}{dt} = 1 - \frac{2n(t)}{N}. \quad (7.3)$$

The solution of the differential equation (7.3) is

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

where we have used the initial condition $n(t=0) = N$. How does the exponential form (7.4) compare to your Monte Carlo results for various values of N ? We can define a *relaxation time* τ as the time it takes $n(t)$ 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.* A simple modification

Modify your program so that each side of the box is chosen with equal probability. A particle is then moved from the side chosen to the other side. If the side chosen does not have a particle in it, 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 $N = 800$. Compare the behavior of $n(t)$ with your predicted behavior, and with the behavior of $n(t)$ found in Problem 7.3. How do the values of \bar{n} and σ^2 compare? Is this variation of the model realistic?

The above probabilistic method for simulating the approach to equilibrium is an example of a *Monte Carlo* method, 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 at $t = 0$, $n = 8$, and $n' = 0$. 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 $\langle \dots \rangle$ 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.

So far we have run each simulation only once. Clearly, running a Monte Carlo simulation only once cannot reproduce the exact enumeration results, because in general, each run gives somewhat different outcomes if we use a different sequence of random numbers. In general, we need to do a Monte Carlo simulation many times and average over the results to obtain meaningful averages. Each run is called a *trial* or a *sample*. How do you know how many trials to use? The answer usually can be obtained empirically by averaging over more and more trials until the average results do not change within the desired level of accuracy.

7.2 Introduction to 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. Now suppose that we want to characterize the motion of a dust particle in a glass of water. We know that as a given dust particle collides with the water molecules, it changes its direction frequently. Its motion appears so erratic that we cannot predict its trajectory after even a small number of collisions. These considerations suggest that a simple model for the trajectory of a dust particle is that it 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 molecule 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 one-dimensional 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 ℓ . At each interval of time 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. After N steps the displacement x of a walker is given by

$$x_N = \sum_{i=1}^N s_i, \quad (7.5)$$

and the displacement squared x^2 is

$$x_N^2 = \left(\sum_{i=1}^N s_i \right)^2, \quad (7.6)$$

where $s_i = \pm\ell$. We can generate one walk of N steps by flipping a coin N times and increasing x by ℓ each time the coin is heads and decreasing x by ℓ each time the coin is tails. For simplicity, we first assume $p = q = \frac{1}{2}$. We expect that if we average over a sufficient number of walks of N

steps, then the average of x_N , denoted by $\langle x_N \rangle$, would be zero. That is, we expect to find as many steps in one direction as in the opposite direction.

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

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

The first sum in (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 $+\ell^2$ and $-\ell^2$ with equal probability, and hence the average of the second term in (7.7) is zero. Because $s_i^2 = \ell^2$ independently of the sign of s_i , the first term in (7.7) equals $\ell^2 N$ for all walks, and hence equals $\ell^2 N$ on the average. We conclude that

$$\langle x_N^2 \rangle = \ell^2 N. \quad (7.8)$$

If the time interval for a step is Δt rather than unity, we should replace N in (7.8) by $N\Delta t$. The result (7.8) can be generalized to two- and three-dimensional walks where each step is a vector \mathbf{s} of constant magnitude, but in a random direction.

The above derivation of the N -dependence of $\langle x_N \rangle$ and $\langle x_N^2 \rangle$ assumes that $p = \frac{1}{2}$. For general p , it is easy to show that $\langle x_N \rangle = (p - q)\ell N$. What is the meaning of this linear dependence on N ? For $p \neq \frac{1}{2}$, it is convenient to consider the dispersion $\langle \Delta x_N^2 \rangle$ defined as

$$\langle \Delta x_N^2 \rangle \equiv \langle (x_N - \langle x_N \rangle)^2 \rangle = \langle x_N^2 \rangle - \langle x_N \rangle^2. \quad (7.9)$$

It is straightforward to show that for any value of p , the N dependence of $\langle \Delta x_N^2 \rangle$ is given by

$$\langle \Delta x_N^2 \rangle = 4pq\ell^2 N. \quad (7.10)$$

What is the N dependence of $\langle x_N^2 \rangle$ for $p \neq q$?

We 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” and averaging over many trials. The implementation of the random walk algorithm is simple, for example,

```

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

```

The more difficult parts of the program are associated with bookkeeping. The walker takes a total of N steps in each trial and the average values of x_N and x_N^2 are computed.

```

// 3/29/01, 8:30 pm
package edu.clarku.sip.chapter7;
import edu.clarku.sip.graphics.*;
import edu.clarku.sip.templates.*;
import java.awt.*;
import javax.swing.*;

```

```

import java.text.NumberFormat;
// A random walk in 1D
public class OneDimensionalWalk implements AnimationModel, Runnable
{
    private int N;                // maximum number of steps in one trial
    private double p = 0.5;       // probability of right step
    private int trials;           // number of trials
    private int steps;            // steps made by walker
    private int x;                // position of walker
    private double xcum;
    private double x2cum;
    private double xbar;
    private double x2bar;
    private Control myControl = new SAnimationControl(this);
    private Thread animationThread;
    private Histogram histogram = new Histogram(); // part of graphics package
    private World2D world = new World2D();
    // inner class to display variable trials, xbar, and x2bar
    private VariablesDisplay variablesDisplay = new VariablesDisplay();
    private TickMarks tickMarks = new TickMarks(); // part of graphics package
    private NumberFormat numberFormat;

    public OneDimensionalWalk()
    {
        numberFormat = NumberFormat.getInstance();
        numberFormat.setMaximumFractionDigits(2);
        world.setXOffset(60);      // argument in pixels
        world.setYOffset(60);
        world.addDrawable(histogram);
        world.addDrawable(tickMarks);
        world.addDrawable(variablesDisplay);
    }

    public void startCalculation()
    {
        N = (int) myControl.getValue("N"); // number of steps in one trial
        p = myControl.getValue("p");      // probability of right step
        steps = 0;
        trials = 0;
        x = 0;                            // initial position of walker
        xcum = 0;                          // accumulate values of x after N steps
        x2cum = 0;
        histogram.reset();
        animationThread = new Thread(this);
        animationThread.start();
    }
}

```

```
public void stopCalculation()
{
    animationThread = null;
}

public void continueCalculation()
{
    animationThread = new Thread(this);
    animationThread.start();
}

public void clear()
{
    histogram.reset();
    world.repaint();
}

public void step()
{
    // move walker one step until number of steps equals N
    while (!moveOneStep()) // keep calling method moveOneStep while it returns false
        ; // do nothing, semicolon needed to make while loop valid

    // set range of world in real coordinates
    double xmin = histogram.getXMin();
    double xmax = histogram.getXMax();
    int ymin = histogram.getYMin();
    int ymax = histogram.getYMax();
    world.setXYMinMax(xmin, xmax, ymin, ymax);
    world.render();
}

// move walker one step, return true if walker has completed N steps
private boolean moveOneStep()
{
    if (p < Math.random())
        x++;
    else
        x--;
    steps++;
    if (steps == N)
    {
        trials++;
        histogram.addPoint(x);
        xcum = xcum + x;
    }
}
```

```
        x2cum = x2cum + x*x;
        xbar = xcum/trials;
        x2bar = x2cum/trials;
        x = 0;
        steps = 0;
        return true;
    }
    return false;
}

public void run()
{
    while (animationThread == Thread.currentThread())
    {
        step();
        try
        {
            Thread.sleep(100);
        }
        catch(InterruptedException e){}
    }
}

public void reset()
{
    myControl.setValue("N", 16);
    myControl.setValue("p", 0.5);
}

private class VariablesDisplay implements Drawable
{
    public void draw(World2D w, Graphics g)
    {
        g.setColor(Color.black);
        int yOffset = (int) w.getYOffset();
        int xOffset = (int) w.getXOffset();
        int width = w.getSize().width;
        g.drawString("trials = " + trials, xOffset, yOffset/2);
        g.drawString("<x> = " + numberFormat.format(xbar), xOffset + width/4,yOffset/2);
        g.drawString("<x^2> = " + numberFormat.format(x2bar), xOffset + width/2,yOffset/2);
    }
}

public static void main(String[] args)
{
    OneDimensionalWalk odw = new OneDimensionalWalk();
```

```

        odw.reset();
    }
}

```

Problem 7.5. Random walks in one dimension

- In class `OneDimensionalWalk` the steps are of unit length so that $\ell = 1$. Use the program to estimate the number of trials needed to obtain $\langle x_N^2 \rangle$ for $N = 10$ steps with an accuracy of approximately 5%. Compare your result to the exact answer (7.8). Approximately how many trials do you need to obtain the same relative accuracy for $N = 40$?
- Is $\langle x_N \rangle$ exactly zero in your simulations? Explain the difference between the analytical result and the results of your simulations.
- How do your results for $\langle x_N \rangle$ and $\langle \Delta x_N^2 \rangle$ change for $p \neq q$? Choose $p = 0.7$ and determine the N dependence of $\langle x_N \rangle$ and $\langle \Delta x_N^2 \rangle$.
- Determine $\langle x_N^2 \rangle$ for $N = 1$ to $N = 5$ by enumerating all the possible walks. 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 $p = \frac{1}{2}$). 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 N and x and compute the various averages exactly.

Class `OneDimensionalWalk` also computes and displays the distribution of values of the displacement x after N steps by using the `Histogram` class listed in Appendix 7A. One way of determining the number of times that the variable x has a certain value is to define a one-dimensional array and let

```

    histogram(x) = histogram(x) + 1;

```

In this case because x takes only integer values, the array index of `histogram` 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 values of the bin number to the number of occurrences corresponding to the bin. This map is easy to implement in this case, but Java defines a general `Hashtable` class that maps bin numbers (keys) to occurrences (values). The `Histogram` class also draws itself.

Problem 7.6. Probability distribution

- 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

$$A e^{-(x-\langle x_N \rangle)^2/(2\langle \Delta x_N^2 \rangle)}, \quad (7.11)$$

where A is a normalization constant related to $\langle \Delta x_N^2 \rangle$. Compare your computed values for $P_N(x)$ to the form (7.11) using $\langle x_N \rangle$ and $\langle \Delta x_N^2 \rangle$ as input.

- c. Determine $\langle x_N^2 \rangle$ for $N = 1$ to $N = 5$ by enumerating all the possible walks. 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 $p = \frac{1}{2}$). 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 N and x and compute the various averages exactly.

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

$$\frac{\partial P(x, t)}{\partial t} = D \frac{\partial^2 P(x, t)}{\partial x^2}, \quad (7.12)$$

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 $\partial^2/\partial x^2$ 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

$$\langle x_N \rangle = 0 \quad (7.13)$$

and

$$\langle x_N^2 \rangle = 2Dt. \quad (7.14)$$

If we compare the form of (7.8) with (7.14), 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 implies that we can approach many problems in two ways. The traditional way is to formulate the problem as a partial differential equation as in (7.12) 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. In [Chapter 12](#) we will consider random walks in many texts and find that it is straightforward to handle various boundary conditions. Think of other situations that can be treated as random walks (see for example, [Section 10.2](#) and [Chapter ??](#).)

7.3 The Poisson Distribution and Nuclear Decay

As we have seen, we often can change the names of several variables 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. 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 $L = 1000$ equal size regions. The probability that a dart hits a given region in any one throw is $p = 1/1000$. If we count the number of darts in the different regions, we would find that most regions are empty, some regions have one dart, and other regions have more than one dart. What is the probability $P(n)$ that a given region has a particular number of darts?

Problem 7.7. Throwing darts

Write a program that simulates the throwing of N darts at random into L regions in a dart board. Throwing a dart at random at the board is equivalent to choosing an integer at random between 1 and L . Determine $H(n)$, the number of regions with n darts. Note that if $H(n) = H_1(n)$ for the first trial, and $H(n) = H_2(n)$ for the second trial, then $H(n) = H_1(n) + H_2(n)$ for both trials. Obtain $H(n)$ for many trials, and then compute the distribution

$$P(n) = \frac{H(n)}{\sum_{n=0}^N H(n)}. \quad (7.15)$$

As an example, choose $N = 50$ and $L = 1000$. Choose the number of trials to be sufficiently large so that you can determine the qualitative form of $P(n)$.

If N is much greater than unity, then p , the probability that a dart strikes a given region, is much less than unity. The conditions $N \gg 1$ and $p \ll 1$ and the independence of the events (the landing of a dart in a particular region) satisfy the requirements for a *Poisson distribution*. The Poisson distribution, $P(n)$, is given by

$$P(n) = \frac{\langle n \rangle^n}{n!} e^{-\langle n \rangle}, \quad (7.16)$$

where n is the number of darts in a given region 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 since $N \gg 1$, we can take the upper limit to be ∞ when it is convenient.

Problem 7.8. Darts and the Poisson distribution

- Write a program to compute $\sum_{n=0}^N P(n)$, $\sum_{n=0}^N nP(n)$, and $\sum_{n=0}^N n^2 P(n)$ using the form (7.16) for $P(n)$. Choose a reasonable value for $\langle n \rangle$. Verify that $P(n)$ in (7.16) is normalized. What is the value of σ for the Poisson distribution?
- Modify the program that you developed for Problem 7.7 to compute $\langle n \rangle$. Choose $N = 50$ and $L = 1000$ and use your measured value of $\langle n \rangle$ as input to compare the Poisson distribution (7.16) to your computed values of $P(n)$. If time permits, use larger values of N and L .
- Choose $L = 100$ and $N = 50$ and redo part (b). Are your results consistent with a Poisson distribution? What happens if $L = N = 50$?

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 into other nuclei, and that 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 2.8.

Because each nucleus is identical, we assume that during any time interval Δt , each nucleus has the same probability 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. Every unstable nucleus is tested during each time interval. Note that for a system of unstable nuclei, there are many events that can happen during each time interval, for example, $0, 1, 2, \dots, n$ 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. Program `nuclear_decay`, listed below, implements the nuclear decay algorithm.

```
PROGRAM nuclear_decay
! simulation of decay of unstable nuclei
DIM ncum(0 to 1000)
CALL initial(NO,p,ntrial,tmax,ncum())
CALL decay(NO,p,ntrial,tmax,ncum())
CALL output(ntrial,tmax,ncum())
END

SUB initial(NO,p,ntrial,tmax,ncum())
RANDOMIZE
INPUT prompt "initial number of unstable nuclei = ": NO
INPUT prompt "decay probability for unstable nucleus = ": p
INPUT prompt "number of time intervals per trial = ": tmax
INPUT prompt "number of trials = ": ntrial
FOR t = 1 to tmax
    LET ncum(t) = 0      ! accumulate number of unstable nuclei
NEXT t
END SUB

SUB decay(NO,p,ntrial,tmax,ncum())
DIM N(5000)
FOR itrial = 1 to ntrial
    FOR i = 1 to NO
        LET N(i) = 1      ! nucleus = 1 if unstable
    NEXT i
    LET ncum(0) = ncum(0) + NO
    LET n_unstable = NO ! # of unstable nuclei
    FOR t = 1 to tmax
```

```

        FOR i = 1 to NO
            IF N(i) = 1 then
                IF rnd <= p then
                    LET N(i) = 0      ! nucleus decays
                    LET n_unstable = n_unstable - 1
                END IF
            END IF
        NEXT i
        LET ncum(t) = ncum(t) + n_unstable  ! accumulate data
    NEXT t
NEXT itrial
END SUB

SUB output(ntrial,tmax,ncum())
    ! print data to file to be read by separate program
    OPEN #1: name "decay.dat", create new, access output
    PRINT #1: "time", "mean number of unstable nuclei"
    FOR t = 0 to tmax
        PRINT #1: t, ncum(t)/ntrial
    NEXT t
    CLOSE #1
END SUB

```

Problem 7.9. Monte Carlo simulation of nuclear decay

- Program `nuclear_decay` does the simulation `ntrial` times and averages the results. Assume that the time interval is one second. Choose `NO = 100` (the initial number of unstable nuclei), `p = 0.01`, `tmax = 100`, and `ntrial = 20`. 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 2.8? What is the value of λ for this value of p ?
- 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 $N \gg 1$ and $p \ll 1$, we expect that $P(n)$, the probability that n nuclei decay during a specified time interval, is a Poisson distribution. Modify Program `nuclear_decay` so that it outputs the probability that n unstable nuclei decay during the first time interval. Choose `NO = 1000`, `p = 0.001`, `tmax = 1`, and `ntrial = 1000`. 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 (7.16) with your measured value of $\langle n \rangle$ as input. Then consider $p = 0.02$.
- Modify Program `nuclear_decay` so that it outputs the probability that n unstable nuclei decay during two time intervals. Choose `NO = 1000`, `p = 0.001`, `tmax = 2`, and `ntrial = 1000`. Compare the probability you obtain with your results from part (b). How do your results change as the time interval becomes larger?
- Increase p for fixed $N = 1000$ 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. Choose the location of the small circle at random. Do a single run and describe the qualitative differences between the visual and/or audio patterns for the situations in parts (a)–(d)? Choose $N \geq 5000$. 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, nuclei can decay simultaneously. How can you improve the visualization to better approximate a real system?

7.4 Problems in Probability

Because most of the questions in this introductory chapter on random processes can be answered by analytical methods, why bother to simulate these processes? One 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 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 these problems in this way help lead you to a pencil and paper solution?

Problem 7.10. The 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 another box that is empty. So 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 at random the lid of one of the two empty boxes. You now have the opportunity to stay with your original choice, or to 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 7.11. 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 7.12. 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.

Team	Won	Lost	Percentage
Oakland	99	63	0.611
Kansas City	92	70	0.568
California	91	71	0.562
Texas	83	79	0.512
Minnesota	80	82	0.494
Seattle	73	89	0.451
Chicago	69	92	0.429

Table 7.1: The American League West standings for 1989.

- b. A player rolls two dice. If the sum of the two dice is 7 or 11, the player wins immediately. If the sum is 2, 3, or 12, the player loses immediately. If the game is neither won nor lost on the first throw, the initial number is either 4, 5, 6, 8, 9, or 10. The player rolls the dice again until she either wins by repeating her initial number or she loses by rolling a 7. Write a program to determine the probability that the player wins this game (a variation of the game of craps).
- c. 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 7.13. The Boys of Summer

Luck plays a large role in the outcome of any baseball season. The American League West standings for 1989 are given in Table 7.1. Suppose that the teams remain unchanged and their probability of winning a particular game was the same as in 1989. Do a simulation to determine the probability that Oakland would lead the division for another season. For simplicity, assume that the teams play only each other.

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 7.14. 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, what does a typical state look like? Are the coins uniformly distributed as in the initial state or are many cells empty? Does the system approach equilibrium? Write 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 if you consider bigger systems or begin with more coins on each cell?

Problem 7.15. 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 cooked 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 one minute 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?

7.5 Method of Least Squares

There is a long way to go from obtaining data to determining the relation that best describes it. For example, in Problem 7.5 we computed the mean square displacement $\langle x_N^2 \rangle$ as a function of N for a simple random walk. In Problem 7.9 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 ? Are our simulation results consistent with the theoretical prediction that $\langle x_N^2 \rangle$ is proportional to N for $p = \frac{1}{2}$?

The approach that we have been using is to plot the computed values of $\langle x_N^2 \rangle$ as a function of N and to rely on our eye to help us draw the curve that best fits the data points. Of course, this graphical 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 curve, 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 using a transparent ruler and fitting the data 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 graphical approach is simple, it does not yield precise fits and we need to use analytical methods also. 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 $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ and that the errors are entirely in the values of y . For convenience, we also assume that the uncertainties in y all have the same magnitude. Our goal is to obtain the best fit to the linear function

$$y = mx + b. \quad (7.17)$$

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

$$d_i = y_i - mx_i - b \quad (7.18)$$

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

$$S = \sum_{i=1}^n (y_i - mx_i - b)^2. \quad (7.19)$$

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, then the values of d_i would be distributed according to the Gaussian distribution (see Problems 7.5 and 12.8). Based on this assumption, it can be shown that the values of m and b that minimize S 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.

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

$$\frac{\partial S}{\partial m} = -2 \sum_{i=1}^n x_i (y_i - mx_i - b) = 0, \quad (7.20a)$$

$$\frac{\partial S}{\partial b} = -2 \sum_{i=1}^n (y_i - mx_i - b) = 0. \quad (7.20b)$$

From (7.20) we obtain two simultaneous equations:

$$m \sum_{i=1}^n x_i^2 + b \sum_{i=1}^n x_i = \sum_{i=1}^n x_i y_i \quad (7.21a)$$

$$m \sum_{i=1}^n x_i + bn = \sum_{i=1}^n y_i. \quad (7.21b)$$

It is convenient to define the average quantities

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad (7.22a)$$

$$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i \quad (7.22b)$$

$$\overline{xy} = \frac{1}{n} \sum_{i=1}^n x_i y_i, \quad (7.22c)$$

and rewrite (7.21a) and (7.21b) as

$$m\overline{x^2} + b\bar{x} = \overline{xy}, \quad (7.23a)$$

$$m\bar{x} + b = \bar{y}. \quad (7.23b)$$

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

Table 7.2: Computed values of the mean square displacement $\langle x_N^2 \rangle$ as a function of the total number of steps N . The results $\langle x_N^2 \rangle$ are 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 solution of (7.23a) and (7.23b) can be expressed as

$$m = \frac{\overline{xy} - \bar{x}\bar{y}}{(\Delta x)^2} \quad (7.24a)$$

$$b = \bar{y} - m\bar{x}. \quad (7.24b)$$

where

$$(\Delta x)^2 = \overline{x^2} - \bar{x}^2 \quad (7.24c)$$

Equations (7.24a)–(7.24c) determine the slope m and the intercept b of the best straight line through the n 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 $\langle x_N^2 \rangle$ obeys the general relation

$$\langle x_N^2 \rangle = aN^{2\nu} \quad (7.25)$$

with an unknown exponent ν . First we convert the nonlinear relation (7.25) to a linear relation by taking the logarithm of both sides:

$$\ln \langle x_N^2 \rangle = \ln a + 2\nu \ln N. \quad (7.26)$$

If we take the logarithm of the data in Table 7.2 and use (7.24), we find that $m = 1.02$ and $b = 0.83$. Hence, we conclude from our limited data and the relation $2\nu = m$ that $\nu \approx 0.51$, a numerical result that is consistent with the expected result $\nu = 1/2$. The values of $y = \ln \langle x_N^2 \rangle$ and $x = \ln N$ and the least squares fit are shown in Figure 7.2. Note that the original fitting problem is nonlinear, that is, $\langle x_N^2 \rangle$ 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.

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

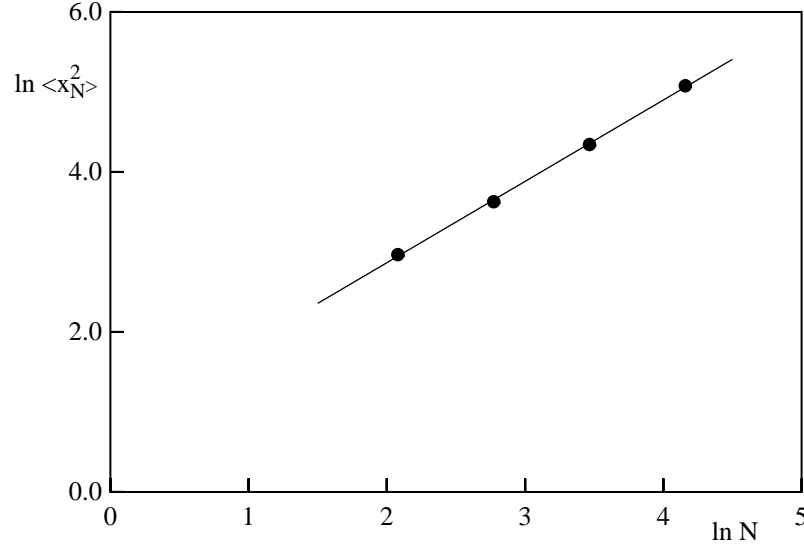


Figure 7.2: Plot of $\ln \langle x_N^2 \rangle$ versus $\ln N$ for the data listed in Table 7.2. The straight line $y = 1.02x + 0.83$ through the points is found by minimizing the sum (7.19).

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

$$\sigma_m = \frac{1}{\sqrt{n}} \frac{\sigma_y}{\Delta x} \quad (7.27a)$$

$$\sigma_b = \frac{1}{\sqrt{n}} \frac{(\overline{x^2})^{1/2}}{\Delta x} \sigma_y, \quad (7.27b)$$

where

$$\sigma_y^2 = \frac{1}{n-2} \sum_{i=1}^n d_i^2, \quad (7.27c)$$

and d_i is given by (7.18). Because there are n data points, we might have guessed that n rather than $n-2$ would be present in the denominator of (7.27c). The reason for the factor of $n-2$ is related to the fact that to determine σ_y , 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 (7.27c), we see that both the numerator and denominator would be zero, and hence $\sigma_y = 0/0$, that is, σ_y is undetermined. If a factor of n appeared in (7.27c) instead, we would conclude that $\sigma_y = 0/2 = 0$, an absurd conclusion. Usually $n \gg 1$, and the difference between n and $n-2$ is negligible.

For our example, $\sigma_y = 0.03$, $\sigma_b = 0.07$, and $\sigma_m = 0.02$. The uncertainties δm and $\delta \nu$ are related by $2\delta \nu = \delta m$. Because we can associate δm with σ_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

$$\chi^2 = \sum_{i=1}^n w_i (y_i - mx_i - b)^2. \quad (7.28)$$

The resulting expressions (7.24a) and (7.24b) for m and b are unchanged if we generalize the definition of the averages to be

$$\bar{f} = \frac{1}{n\bar{w}} \sum_{i=1}^n w_i f_i, \quad (7.29)$$

where

$$\bar{w} = \frac{1}{n} \sum_{i=1}^n w_i \quad (7.30)$$

In Chapter 11 we discuss how to estimate the most probable errors in $\langle x_N^2 \rangle$.

Problem 7.16. Example of least squares fit

- 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.
- 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 $\langle x_N^2 \rangle$ as done in the text. Is your most probable estimate for ν closer to $\nu = 1/2$?

For the simple random walk problems considered in this chapter, the relation $\langle x_N^2 \rangle = aN^\nu$ holds for all N . However, in many random walk problems (see Chapter 12), a power law relation between $\langle x_N^2 \rangle$ and N holds only asymptotically for large N , and hence we should use only the larger values of N to estimate the slope. We also need to give equal weight to all intervals of the independent variable N . In the above example, we used $N = 8, 16, 32$, and 64 , so that the values of $\ln N$ are equally spaced.

7.6 A Simple Variational Monte Carlo Method

Many problems in physics can be formulated in terms of a variational principle. In the following, we consider examples of variational principles from geometrical optics and classical mechanics. We then discuss how Monte Carlo methods can be applied to obtain estimates for the maximum or minimum. A more sophisticated application of Monte Carlo methods to a variational problem in quantum mechanics is discussed in Chapter ???. 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 a principle due to Fermat:

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:

$$v = \frac{c}{n}. \quad (7.31)$$

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 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 **Program fermat**, is to begin with an arbitrary 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 **Program fermat** include:

- a. Light propagates from left to right through N media.
- b. The width of each region is unity and the index of refraction is uniform in each region. The index i increases from left to right. There are $N - 1$ boundaries separating the N media with index of refraction $n(i)$ and speed $v(i)$. We have chosen units such that the speed of light in a vacuum equals unity.
- c. 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.
- d. The coordinates of the light source and the detector are at $(1, y(1))$ and $(N, y(N))$ respectively, where $y(1)$ and $y(N)$ are fixed.
- e. The initial path is the connection of the set of random points at the boundary of each region.
- f. 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 $-\delta$ to δ . If the trial value of $y(i)$ yields a shorter travel time, this value becomes the new value for $y(i)$.
- g. The path is redrawn whenever it is changed.

```
PROGRAM fermat
! Monte Carlo method for finding minimum optical path
DIM y(101),v(101)
CALL initial(y(),v(),N)
CALL change_path(y(),v(),N)
END
```

```

SUB initial(y(),v(),N)
  RANDOMIZE
  LET N = 10                ! number of different media (even)
  ! speed of light in vacuum equal to unity
  LET v1 = 1.0
  LET index = 1.5          ! index of refraction of medium #2
  LET v2 = 1/index
  FOR i = 1 to N/2
    LET v(i) = v1          ! speed of light in left half
  NEXT i
  FOR i = N/2 + 1 to N
    LET v(i) = v2          ! speed of light in right half
  NEXT i
  LET y(1) = 2              ! fixed source
  LET y(N) = 8              ! fixed detector
  SET WINDOW 0.5,N+1,y(1)-0.5,y(N)+0.5
  BOX LINES 1,N,y(1),y(N)
  PLOT LINES: N/2,y(1);N/2,y(N) ! boundary
  ! choose initial path at boundary between endpoints
  FOR i = 2 to N-1
    LET y(i) = (y(N) - y(1))*rnd + y(1)
  NEXT i
  SET COLOR "red"
  FOR i = 1 to N-1
    PLOT LINES: i,y(i);i+1,y(i+1) ! initial path
  NEXT i
  SET COLOR "red/white"
END SUB

SUB change_path(y(),v(),N)
  LET delta = 0.5          ! maximum change in y
  DO
    ! choose random x not including x = 1 or N
    LET x = int((N-2)*rnd) + 2
    LET ytrial = y(x) + (2*rnd - 1)*delta ! new y position
    LET dy2 = (y(x) - y(x+1))^2 ! vertical distance squared
    LET dist = sqr(1 + dy2) ! horizontal distance is unity
    LET t_original = dist/v(x+1)
    LET dy2 = (y(x) - y(x-1))^2
    LET dist = sqr(1 + dy2)
    LET t_original = t_original + dist/v(x)
    LET dy2 = (ytrial - y(x+1))^2
    LET dist = sqr(1 + dy2)
    LET t_trial = dist/v(x+1)
    LET dy2 = (ytrial - y(x-1))^2

```

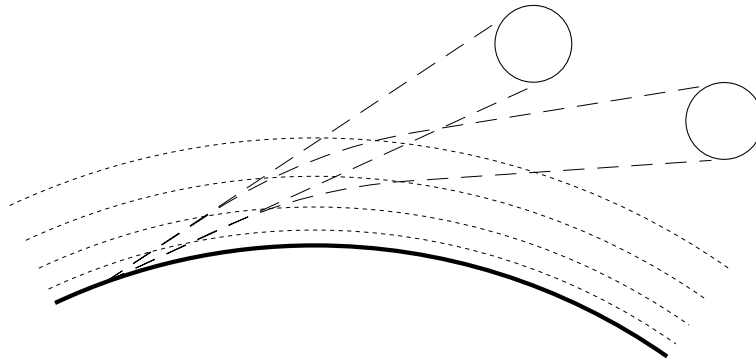


Figure 7.3: Near the horizon, the apparent (exaggerated) position of the sun is higher than the true position of the sun. Note that the light rays from the true sun are curved due to refraction.

```

    LET dist = sqr(1 + dy2)
    LET t_trial = t_trial + dist/v(x)
    IF t_trial < t_original then      ! new position reduces time
      SET COLOR "white"
      PLOT LINES: x-1,y(x-1);x,y(x);x+1,y(x+1)
      SET COLOR "red"
      LET y(x) = ytrial
      PLOT LINES: x-1,y(x-1);x,y(x);x+1,y(x+1)
    END IF
  LOOP until key input
END SUB

```

Problem 7.17. The law of refraction

- Use **Program fermat** 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 $N = 10$ 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 some 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 $(1, y(1))$ and $(10, y(10))$.
- 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 in (a) and (b) are consistent with Snell's law, $n_2 \sin \theta_2 = n_1 \sin \theta_1$.

Problem 7.18. Inhomogeneous media

- 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. Take the index of refraction of region i to be $n(i) = 1 + (i - 1) \cdot dn$. Run Program `fermat` 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 7.3).

- * Use Program `fermat` to find the appropriate distribution of $n(i)$ for a fiber optic cable. In this case the i th 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. Imagine the cable to be a flat, long ribbon of width equal to N . The middle region is the center of the cable and the $i = 1$ and $i = N$ regions are at the edge of the cable. 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 :

$$S = \int_{t_{\text{initial}}}^{t_{\text{final}}} L dt \quad (7.32)$$

The Lagrangian L in (7.32) 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 (7.32), we would find that the path for which S is an extremum satisfies the differential equation of motion equivalent to 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 ??).

Our main motivation here is to gain more experience with extremum problems. To use (7.32) to find the motion of a single particle in one dimension, we fix the position at the initial and final times, $x(t_{\text{initial}})$ and $x(t_{\text{final}})$, and then choose the velocities and positions for other times $t_{\text{initial}} < t < t_{\text{final}}$ so as to minimize the action. One way to implement this procedure numerically is to convert the integral in (7.32) to a sum:

$$S \approx \sum_{i=1}^{N-1} L(t_i) \Delta t, \quad (7.33)$$

where $t_i = t_{\text{initial}} + i\Delta t$. (The approximation used to obtain (7.33) is known as the rectangular approximation and is discussed in Chapter 11.) For a single particle in one dimension, we can write

$$L_i \approx \frac{m}{2(\Delta t)^2} (x_{i+1} - x_i)^2 - u(x_i), \quad (7.34)$$

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 7.19.* Principle of least action

- a. Write a program to minimize the action S given in (7.32) for the motion of a single particle in one dimension. Use the approximate form of the Lagrangian given in (7.34). One way to write the program is to modify *Program fermat* so that the vertical coordinate for the light ray becomes the position of the particle, and the horizontal region number i of width Δx becomes the discrete time interval number of duration Δt . The quantity to be minimized is different, but otherwise the algorithm is similar. 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.
- b. Verify your program for the case of free fall for which the potential energy is $u(x) = mgx$. Choose $x(t = 0) = 2$ m and $x(t = 10 \text{ s}) = 8$ m, and begin with $N = 20$. Allow the maximum change in the position to be 5 m. Make sure that the window coordinates are sufficiently large to see the paths. What do you expect the shape of the plot of x versus t to be?
- 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.

Appendix 7A: Listing of Histogram Class

```

package edu.clarku.sip.graphics;
import java.awt.*;
import java.util.*;

public class Histogram implements Drawable
{
    private Hashtable bins = new Hashtable(); // Hashtable maps bin number to occurrences
    private double binWidth = 1; // 0 < 1, 1 < 2, 2 < 3, etc.
    private boolean discrete = true; // false if the bins are continuous
    private double xmin, xmax;
    private int ymin, ymax;
    private Color binColor = Color.red;

    public Histogram()
    {
        resetXYMinMax();
    }

    public void setDiscrete(boolean b)
    {
        discrete = b;
    }

    public void setBinColor(Color c){binColor = c;}

    private void resetXYMinMax()

```

```
{
    xmin = Integer.MAX_VALUE;
    xmax = Integer.MIN_VALUE;
    ymax = 10;
    ymin = 0;
}

public void setBinWidth(double d)
{
    binWidth = d;
}

public double getXMin(){return xmin;}
public double getXMax(){return xmax;}
public int getYMin(){return ymin;}
public int getYMax(){return ymax;}

public void addPoint(double value)
{
    int binNumber = (int)(value/binWidth);
    // determine if there has been any occurrences for this bin
    Integer occurrences = (Integer) bins.get(new Integer(binNumber));
    int count = 1;
    if (occurrences == null)
    {
        bins.put(new Integer(binNumber), new Integer(count)); // binNumber has one occurrence
    }
    else
    {
        // need to put objects in hashtable, but can only add ints
        count = count + occurrences.intValue(); // increase occurrences by one
        bins.put(new Integer(binNumber), new Integer(count));
    }
    ymax = Math.max(count, ymax);
    xmin = Math.min(binNumber - binWidth, xmin);
    xmax = Math.max(binNumber + binWidth, xmax);
}

public void draw(World2D p, Graphics g)
{
    if (bins.size() == 0)
        return;

    drawHistogram(p, g);
}
```

```

public void drawHistogram(World2D p, Graphics g)
{
    Hashtable temp = (Hashtable) bins.clone();
    Enumeration keys = temp.keys();
    g.setColor(Color.black);
    int xoffset = (int) p.getXOffset();
    int yoffset = (int) p.getYOffset();
    // draw an enclosing box
    g.drawRect(xoffset, yoffset, p.getSize().width - xoffset*2, p.getSize().height - 2*yoffset);
    g.setColor(binColor);
    while (keys.hasMoreElements())
    {
        Integer binNumber = (Integer) keys.nextElement();
        Integer occurrences = (Integer) bins.get(binNumber);
        drawBin(p, g, binNumber.intValue(), occurrences.intValue());
    }
}

public void drawBin(World2D p, Graphics g, int binNumber, int occurrences)
{
    int px = p.xToPix(binNumber*binWidth);
    if (discrete)
    {
        g.drawLine(px, p.yToPix(0), px, p.yToPix(occurrences));
    }
    else // continuous distribution
    {
        int py = p.yToPix(occurrences);
        int px2 = p.xToPix(binNumber*binWidth + binWidth);
        int pWidth = px2 - px;
        int pHeight = Math.abs(py - p.yToPix(0));
        g.fillRect(px, py, pWidth, pHeight);
    }
}

public void reset()
{
    bins = new Hashtable();
    resetXYMinMax();
}

public static void main(String[] args)
{
    Histogram h = new Histogram();
    h.setDiscrete(false);
    h.addPoint(1);
}

```

```

    h.addPoint(1);
    h.addPoint(3);
    h.addPoint(-3);
    World2D w = new World2D();
    w.setXYMinMax(-4,4,0,3);
    w.setXOffset(60);
    w.setYOffset(60);
    w.addDrawable(h);
    w.addDrawable(new TickMarks());
    w.repaint();
}
}

```

Appendix 7B: Random Walks and the Diffusion Equation

To gain some insight into the relation between random walks and the diffusion equation, we 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 (7.12) here for convenience:

$$\frac{\partial P(x, t)}{\partial t} = D \frac{\partial^2 P(x, t)}{\partial x^2}, \quad (7.35)$$

To derive the t dependence of $\langle x(t) \rangle$ and $\langle x^2(t) \rangle$ from (7.35), we write the average of any function of x as

$$\langle f(x, t) \rangle = \int_{-\infty}^{\infty} f(x) P(x, t) dx. \quad (7.36)$$

The average displacement is given by

$$\langle x(t) \rangle = \int_{-\infty}^{\infty} x P(x, t) dx. \quad (7.37)$$

To do the integral on the right hand side of (7.37), we multiply both sides of (7.35) by x and formally integrate over x :

$$\int_{-\infty}^{\infty} x \frac{\partial P(x, t)}{\partial t} dx = D \int_{-\infty}^{\infty} x \frac{\partial^2 P(x, t)}{\partial x^2} dx. \quad (7.38)$$

The left-hand side can be expressed as:

$$\int_{-\infty}^{\infty} x \frac{\partial P(x, t)}{\partial t} dx = \frac{\partial}{\partial t} \int_{-\infty}^{\infty} x P(x, t) dx = \frac{d}{dt} \langle x \rangle. \quad (7.39)$$

The right-hand side of (7.38) can be written in the desired form by doing an integration by parts:

$$D \int_{-\infty}^{\infty} x \frac{\partial^2 P(x, t)}{\partial x^2} dx = D x \frac{\partial P(x, t)}{\partial x} \Big|_{x=-\infty}^{x=\infty} - D \int_{-\infty}^{\infty} \frac{\partial P(x, t)}{\partial x} dx. \quad (7.40)$$

The first term on the right hand side of (7.40) 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

$$\frac{d}{dt}\langle x \rangle = 0, \quad (7.41)$$

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

$$\frac{d}{dt}\langle x^2(t) \rangle = 2D, \quad (7.42)$$

or

$$\langle x^2(t) \rangle = 2Dt. \quad (7.43)$$

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

References and Suggestions for Further Reading

- 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.
- Philip R. Bevington and D. Keith Robinson, *Data Reduction and Error Analysis for the Physical Sciences*, second edition, McGraw-Hill (1992).
- 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.
- A. K. Dewdney, "Computer Recreations (Five easy pieces for a do loop and random-number generator)," *Sci. Amer.* **252**(#4), 20 (1985).
- Robert M. Eisberg, *Applied Mathematical Physics with Programmable Pocket Calculators*, McGraw-Hill (1976). Chapter 7 discusses entropy and the arrow of time.
- 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.
- Peter R. Keller and Mary M. Keller, *Visual Cues*, IEEE Press (1993). A well illustrated book on data visualization techniques.
- 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.

- F. Reif, *Statistical and Thermal Physics*, Berkeley Physics, Vol. 5, McGraw-Hill (1965). Chapter 2 introduces random walks.
- 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.
- 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. Tufte, *The Visual Display of Quantitative Information*, Graphics Press (1983).
- 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).