Ronald W. Shonkwiler
Franklin Mendivil

UNDERGRADUATE TEXTS IN MATHEMATICS

Explorations in Monte Carlo Methods

$$\begin{tikzcd}
 & x_1 \arrow[loop above, "p_{11}"] \arrow[dl, "p_{14}"'] \arrow[dr, bend right, "p_{12}"'] & \\
x_4 \arrow[loop left, "p_{44}"] \arrow[dr, bend right, "p_{43}"'] & & x_2 \arrow[ul, bend right, "p_{21}"'] \arrow[dl, "p_{23}"] \\
 & x_3 \arrow[ul, bend right, "p_{34}"'] \arrow[uu, "p_{31}"'] &
\end{tikzcd}$$

Springer

Ronald W. Shonkwiler
Franklin Mendivil

Explorations in Monte Carlo Methods

Springer

Ronald W. Shonkwiler
School of Mathematics
Georgia Institute of Technology
Atlanta, GA 30332-0160
USA
shonkwiler@math.gatech.edu

Franklin Mendivil
Department of Mathematics and Statistics
Acadia University
Wolfville, NS B4P 2R6
Canada
franklin.mendivil@acadiau.ca

ISSN 0172-6056
ISBN 978-0-387-87836-2 e-ISBN 978-0-387-87837-9
DOI 10.1007/978-0-387-87837-9
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2009932312

Mathematics Subject Classification (2000): 65C05 (primary), 68T05, 60J20 (secondary), 60G40 (tertiary)

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The Monte Carlo method is a technique for analyzing phenomena by means of computer algorithms that employ, in an essential way, the generation of random numbers.

A solution by Monte Carlo methods was one of the very first uses made of the newly invented digital computer. In a very real sense the method was "born" with the computer. From the start, Monte Carlo methods have been used to solve difficult problems for which no other solution method was available at the time. This is still the case. In some cases better methods arose and displaced Monte Carlo; as it should be. Yet, in many applications Monte Carlo is unsurpassed. It still enjoys almost exclusive dominion over its original application, simulating complex interactions in any area where quantitative models are possible. In the meantime Monte Carlo has moved ahead as well, finding new areas of application and enjoying new resurgence in former areas as a result of increased computer power. Today Monte Carlo methods are more widespread than ever.

Monte Carlo methods originated in their modern form with, and were named by, Stanislaw Ulam and John von Neumann. Later Ulam went on to expand on the method and champion its use. He inspired many subsequent adherent's who themselves developed and extended the method. To "Stan," as he was known to his friends, Monte Carlo was just one technique for performing mathematical experiments on the computer, an idea in which he fervently believed.

In this book we show how Monte Carlo can be used to solve problems in science, engineering, business, and industry. But most of all, we intend to have fun. We will use the computer to explore the hidden and sometimes surprising nature of quantitative systems. From throwing needles on cracks to playing blackjack, from following sea birds searching for food to using computer creatures to solve optimization problems, we will explore what Monte Carlo can do.

Along the way we will learn and improve skills in probability, statistics, programming, mathematics, and even Monte Carlo methods. For example, the central limit theorem plays a central role in many parts of the book. And we will learn that computer random numbers are not random at all; nevertheless, a source of good “pseudorandom” numbers is essential to the method.

The initial mathematical skills the student should have are calculus through integration and matrix arithmetic. A familiarity with programming is helpful, but the programming in this text starts at an elementary level and proceeds gradually in scope. A “computer inclined” mind is probably more important. Our demonstration programs are amply commented.

Programming is at the heart of Monte Carlo methods and will be the primary activity of our work in this text. All that we learn and discover emerges in the display of the results of our programs. Toward that end we provide a large number of problems for computer solution at the end of each chapter. The statement of each problem is prefixed by a number in parentheses indicating the relative difficulty or time requirement for that problem. These are only estimates, however; your perception may be different, but hopefully not by too much. Keep in mind that creating and debugging code always takes more time than expected. Hofstadter’s Law states: it always takes longer than you expect, even when you take Hofstadter’s Law into account.

With regard to programming, quick, short, and simple solutions is a major feature of Monte Carlo methods. For example the program on page 4 for simulating the Buffon needle problem mentioned above is just seven lines. Generally, programs in the earlier chapters tend to be small and simple. Later on, they become a little more elaborate. Thus the program for pricing options via simulation on page 178 is a bit longer at 24 lines.

Besides the numerical computations, the results have to be graphically displayed. Most convenient is a software package that can do both. We have chosen to use Matlab for illustration purposes throughout. Rather than using pseudocode for this purpose, Matlab is itself easy to understand, and the code presented can be directly run within Matlab. The included code produced many of the figures in the text. However, while Matlab is good as a mathematics and graphics package, it is not optimal as a programming language. We recommend that for the serious implementation of a Monte Carlo solution, for example in optimization, a richer programming language be used such as C or Java.

As a note to any instructor using this book, we have found over the years in teaching a course based on this material that an excellent way to finish up is having students do a final project of their own choosing; see Appendix C. Be prepared for some very impressive submissions.

Atlanta, Georgia *Ronald Shonkwiler*
Wolfville, Nova Scotia *Franklin Mendivil*

Acknowledgments

One of us (Shonkwiler) had the great fortune of having gotten to know Stan Ulam, in part through a set theory course he taught at the University of Colorado. As anyone whose life he touched will testify, he was a most enthusiastic, inspiring, genuinely creative, and yet humble person.

We wish to thank David Kramer for his outstanding job copyediting the original manuscript, Laura Held for her cheerful management of the work from its infancy to completion, and Ann Kostant for her initial inspiration and for looking in from time to time keeping us on track. Finally, we give a special thanks to our students, who show an abiding interest in the material, ask insightful questions, and sometimes find novel solutions to the problems. Ultimately it is for them that our effort is sustained.

Contents

1
Introduction to Monte Carlo Methods

1.1 How Can Random Numbers Solve Problems?

The Monte Carlo method is a technique for analyzing phenomena by means of computer algorithms that employ, in an essential way, the generation of random numbers. The Monte Carlo method was given its name by Stanislaw Ulam and John von Neumann, who invented the method to solve neutron diffusion problems at Los Alamos in the mid 1940s.

Monte Carlo methods are widely used in mathematics, science, industry, commerce, and entertainment. They are at the heart of algorithms used to make predictions about stochastic processes, that is, phenomena having some random component. This includes the motion of microscopic particles in an environment, the generation and movement of data packets through networks, the arrival and servicing of vessels at a busy port, and hundreds of other processes about which people need answers. Random numbers are used directly in the transmission and security of data over the airwaves or along the Internet. A radio transmitter and receiver could switch transmission frequencies from moment to moment, seemingly at random, but nevertheless in synchrony with each other. The Internet data could be credit-card information for a consumer purchase, or a stock or banking transaction secured by the clever application of random numbers. And randomness is an essential ingredient in games of all sorts, computer or otherwise, to make for unexpected action and keen interest.

1.1.1 History of the Monte Carlo Method

However, the first application of the type of analysis that would later be called the Monte Carlo method was employed by the French naturalist Comte de Buffon (1707–1788) in 1733. He imagined an experiment for estimating the value of $\pi = 3.1415926\ldots$ by tossing a needle onto a ruled surface such as that produced by hardwood flooring. Let us get started in Monte Carlo methods by simulating his needle problem.

Buffon Needle Problem

Consider a needle of length L thrown at random onto a flat surface ruled by parallel lines each a distance d apart. Let H_n be the random variable counting the number of times the needle touches or crosses a line in n throws, the number of "hits."

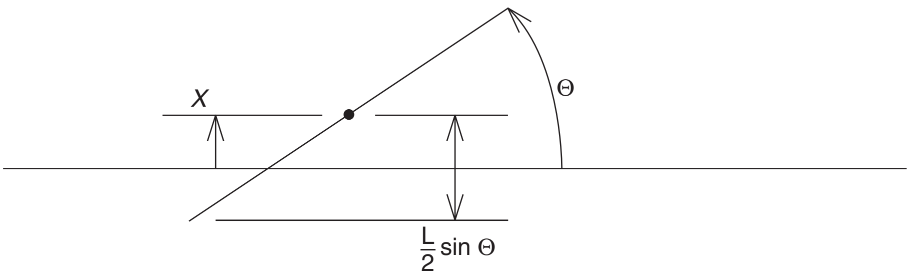


Fig. 1.1. Buffon needle on a crack.

Let X denote the distance the center of the needle lands from the nearest line and Θ the acute angle the needle makes with respect to the ruled lines; see Figure 1.1. Then $0 \le X \le \frac{d}{2}$ and $0 \le \Theta \le \frac{\pi}{2}$. The needle touches or crosses a line if and only if

$$X \le \frac{L}{2}\sin\Theta.$$

Graphically, the probability of a hit is equal to the area under the curve $y = \frac{L}{2}\sin\Theta$. If $d \ge L$, then

$$\begin{aligned}\Pr(\text{needle crosses a line}) &= \frac{\text{area under the curve}}{\text{area of rectangle } (0,\pi/2)\times(0,d/2)}\\ &= \frac{\int_0^{\pi/2}\frac{L}{2}\sin\Theta\,d\Theta}{\frac{d}{2}\frac{\pi}{2}} = \frac{2L}{\pi d}.\end{aligned}$$

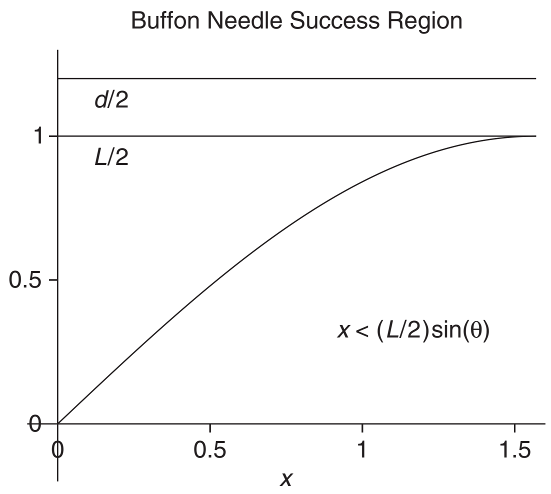


Fig. 1.2. Buffon needle "dartboard."

Take $d = 2L$, for example; then the probability of a hit is $1/\pi$.

In performing the actual experiment, with a real needle and a hardwood floor, the value of π is not known in advance; rather, an estimate for it emerges from the experiment. What was remarkable about Buffon's solution is that no one had previously dreamed that π could be determined by a random experiment! Unfortunately, as we will see on page 39, the accuracy of the estimate only improves in proportion to $1/\sqrt{n}$, where n is the number of tosses of the needle, that is, trials of the experiment. A very large number of trials is needed to get even modest accuracy; see Figure 1.13. But this obstacle is ameliorated by the digital computer and its ability to quickly simulate many trials.

Just as envisioned by Buffon, scientists at Los Alamos in 1943 sought to solve their neutron flux problem by a statistical experiment. But they could not perform the actual experiment; it had to be simulated. They realized that they could produce the same statistical results of an actual experiment through the use of a mathematical model. The only real experiment needed was the generation of truly random numbers, and a large number of them. From there, the model converted the random numbers into the simulated physics.

Initially, the entire simulation was done manually by dozens of workers using mechanical desktop calculators. But von Neumann, a consultant on the Manhattan Project, was aware of and involved in the development of the programmable digital computer. It was immediately realized that the computer was exactly the tool needed for these simulations, thereby saving weeks of manual labor. Of course, there was still the problem of generating the required random numbers. In fact, von Neumann proposed solving this problem in an insightful way, by the use of the digital computer itself, and created the first pseudorandom number generator, the "middle-square" generator.

Following this lead, we will perform Buffon's needle experiment by computer simulation. A convenient choice for the parameters is $d = 2$ and $L = 1$. The program generates two random numbers, one to play the role of X, $0 \le X < d/2$, and the other of Θ, $0 \le \Theta < \pi/2$. Next we calculate whether $X \le \frac{L}{2}\sin\Theta$ and record a hit if so. Repeating this n times, π is estimated as n/H_n. For now, take $n = 10{,}000$.

Keep in mind that the purpose of this exercise is to get started in Monte Carlo methods and not to determine π, a value we already know. In fact, we need the value of π to generate the random values of Θ needed in our Buffon needle simulation. For a Monte Carlo simulation that estimates π ab initio, see Section 1.4.

Here is the program written in Matlab. We rely on Matlab for the random number generation and the comparison. Matlab can also graph the results in the form of a histogram. Histogramming is one of our main analytical tools and we will discuss it shortly.

```
 Matlab
> throws = 10000; % percent makes the rest of the line a comment
> x=rand(1,throws); % a vector of 10000 pseudorandom numbers in the range [0,1)
> theta=rand(1,throws); % semicolon supresses printing
> theta=0.5*pi*theta; % theta now 10000 random numbers between 0 and π/2
> hits= x<=0.5*sin(theta); % hits is a vector of 0's where false and 1's where true
> sum(hits) % no semicolon prints the number of hits out of the 10000 throws
> piEst=throws/sum(hits)
```

1.1.2 Histogramming Simulation Results

In running our Buffon needle simulation we obtain a single number, an estimate for π. Is it too high, too low, close to the correct value, how close? We don't know (continuing our assumption that π is not known in advance). But there is a way to answer these questions; it is by running multiple experiments and histogramming the results.

A histogram is a special kind of bar chart for depicting a series of values, v_1, v_2, …, v_n, numbering, say, n in total. A convenient subdivision of the x-axis is created containing the values, for example by means of the points $x_0, x_1, x_2, \ldots, x_K$. They establish intervals, or *bins*,

$[x_0, x_1)$, $[x_1, x_2)$, …, $[x_{K-1}, x_K)$. A count is made of how many of the values lie in each bin and a bar of that height is depicted standing on that bin. Technically this is called a *frequency histogram.* Dividing the frequencies by n, the total number of values, calculates the *relative frequencies*. Making the heights equal to these values generates a *density histogram.* A density histogram is an approximation to the probabilistic distribution of the process the values represent. We take up the idea of probability distribution in the next section.

Let us modify our Buffon needle program to produce a histogram. For technical reasons, this time we make an estimate of $1/\pi$. The reason has to do with the fact that we want the random variable to be in the numerator rather than the denominator. Therefore we estimate $1/\pi = 0.318309\ldots$ using 100 trials of the basic experiment. But in addition, consider an experiment consisting of these basic experiments, let's call it the "super experiment." We perform 2000 such super experiments and histogram these data. Altogether we will be performing two hundred thousand simulated needle tosses. The resulting histogram can be used to provide probabilistic accuracy statements about the estimated value of π.

Figure 1.3 shows the resulting histogram produced by the Matlab program below. We see that it has the bell shape of a normal distribution, a topic we take up in Section 2.6. The average of the 2000 super experiments is an approximation of the *mean* of the distribution and is likewise an approximation of $1/\pi$. For the run shown in Figure 1.3 the average is $\bar{p} = 0.3180$.

The width of a normal distribution is determined by a parameter called the *standard deviation.* The standard deviation for the run shown is $s = 0.0467$. A property of the normal distribution is that the probability that a sample drawn from it falls between two standard deviations on either side of the mean is 0.954. Hence a single sample usually lies between $0.3180-2\cdot 0.0467$ and $0.3180+2\cdot 0.0467$. For the average of 2000 samples, the interval shrinks to $2\cdot 0.0467/\sqrt{2000} = 0.0021$ on either side of the mean. Finally, since an interval centered on the true mean will contain a sample mean if and only if the same interval centered on the sample mean contains the true mean, we can say

$$\text{probability that } (0.3180 - 0.0021 < 1/\pi < 0.3180 + 0.0021) = 0.954.$$

In other words, only about 5% of the time will this experiment produce an interval of width $0.0042 = 2 \cdot 0.0021$ that fails to contain $1/\pi$.

For π itself, just reciprocate the inequalities,

$$\text{probability that } (1/0.3201 < \pi < 1/0.3159) = 0.954.$$

In the example at hand, the interval falls between 3.124 and 3.165. For more on all this see Section 2.6.3.

```
  Matlab
> throws = 100; % 100 trials
> replications=2000;
> for i=1:replications
    x=rand(1,throws);
    theta=0.5*pi*rand(1,throws);
    hits= x<=0.5*sin(theta);
    y(i)=sum(hits)/throws; % y= result vector
> end
> hist(y) % let hist pick the bins
> recipEst=sum(y)/replications
> vVec=(y-recipEst).*(y-recipEst); % vector of sqr deviations
> v=sum(vVec)/(replications-1); % sample variance
> stddev=sqrt(v) % sample standard deviation
> piEst=1/recipEst
```

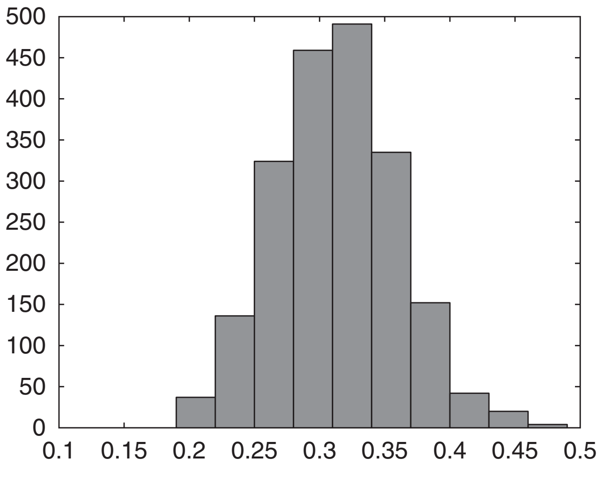


Fig. 1.3. Buffon needle histogram for $1/\pi$.

Histogramming Is an Important Analytical Tool

Histogramming is the main technique we use to understand the results of a Monte Carlo experiment. For example, consider the experiment of rolling a pair of dice. By conducting the experiment one time we get an *instance* of the experiment, that is, one of the possible outcomes. Running the experiment one time is called a *trial* of the experiment. The first step in creating a histogram is performing a large number of such trials and recording the *frequency counts* of the resulting data. In such a dice-rolling experiment one would naturally keep a count of the number of times the outcome was 2 and 3 and 4 and so on up to 12. Then dividing the frequency counts by the total number of trials, one gets an estimate of the probabilities of the outcome for each of those values. Histogramming this dice-rolling experiment is done by plotting the frequency counts as bars of width equal to 1 centered on the possible outcomes, that is, on $2, 3, \ldots, 12$; see Figure 1.4. Alternatively, the frequency counts divided by the number of trials may be plotted instead; producing a density histogram. Then the area under the sequence of bars will sum to 1.

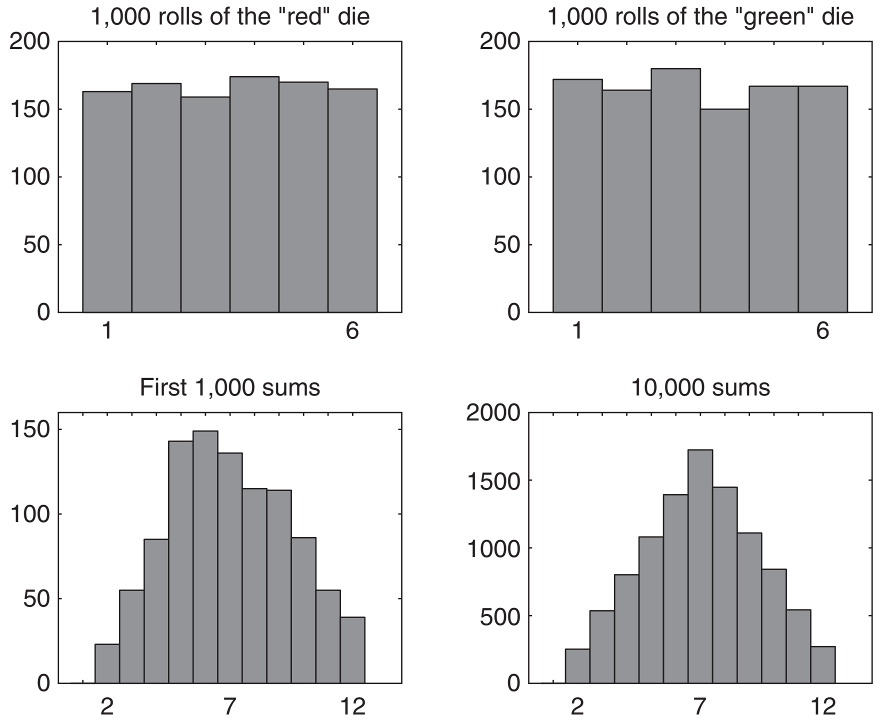


Fig. 1.4. Dice-rolling histograms.

In the upper left and right of Figure 1.4 are plotted the histograms of 1,000 simulated rolls of a “red” and a “green” die. One sees that the outcomes are about evenly distributed among the numbers 1 through 6. The histogram of their sum is plotted in the lower left. As one can see, even for 1,00 trials, the results can be far from the expected probabilities. The histogram for 10,000 trials is plotted in the lower right. These are much closer to the expected results.

Sometimes the outcomes of an experiment do not fall on nice integer values; for example, consider measuring the circumference of hen’s eggs produced by the chickens on a farm. In such a case, we must artificially create a subdivision of the range of possible outcomes, the bins. Now for each instance of the experiment we must determine into which bin the outcome falls and add 1 to the frequency count for that bin. Looking back on the dice-rolling experiment, we see that we did exactly the same thing, only the bins were naturally arranged by the discrete nature of the outcomes.

After conducting a large number of trials and recording the bin frequencies, we depict the results as before by plotting bars lying over each bin of height equal to its frequency. This time the width of the bar will be that of the bin. To obtain a density histogram, the total area under the bars must sum to 1. Therefore the frequencies must be scaled, each multiplied by a constant c, to satisfy the equation

$$1 = \sum_{i=1}^{K} c f_i(\Delta_i x),$$

where K is the total number of bins, f_i is the frequency count of the ith bin, and $\Delta_i x$ is the ith bin width. Solving for c, the frequencies must be scaled by

$$c = \frac{1}{\sum_1^K f_i(\Delta_i x)}.$$

1.1.3 Sample Paths

Another method of portraying the results of a run of an experiment is to plot its *sample path*. Suppose the outcome on the ith trial of the experiment is R_i. The sample path is a plot of the pairs (i, R_i) for $i = 1, 2, \ldots, n$, where n is the number of trials.

To illustrate, consider a gambling experiment in which a gambler makes \$1 bets against the house. Suppose the winning chances are even, that is, the gambler has a 50% chance of winning each bet versus a 50% chance of losing, but the house has much more money than the gambler, say 20 times as much. The gambler starts with $100 and the house with $2,000, and they play until one or the other is bankrupt. We could ask many questions about this experiment, such as; What is the chance the gambler will bankrupt the house? How long will play last until one or the other is bankrupt? and so on.

The following code simulates the experiment. Again we use the built-in random number generator, which returns numbers in the range $[0, 1)$ uniformly. The gambler wins if the random number is less than 0.5; otherwise, the gambler loses. The built-in plotting tools are used to create a sample path of the gambler's fortune for the first trial; Figure 1.5a. Then 20 further trials are run and the playing time T is histogrammed; Figure 1.5b. Note: there is no upper bound to the playing time length; as can be seen from the histogram, a run can take a very long time, e.g., 1,500,000 plays. It may be prudent to add an upper bound to the while loop but unfortunately this will also skew the results (see Problem 9 at the end of this chapter for some more discussion of this).

```
 Matlab
> R=zeros(1,2000000); % vector of 2,000,000 zeros
> i=1; R(i)=100; % gambler's initial fortune
> while( R(i)>0 &R(i)<2100 )
  i=i+1;
  W = (rand <0.5); % random value of 0 or 1
  W = 2*W - 1; % random value of +/-1
  R(i)= R(i-1)+W; % gamblers new fortune
 end
> plot(1:length(R),R) % plot R against its index
>  %%%  run for investigating running time
> for j=1:20
> i=1; R(i)=100;
> while( R(i)>0 &R(i)<2100 )
  i=i+1;
  W = 2*(rand <0.5)-1;
  R(i)= R(i-1)+W;
 end
> T(j)=i
> end
> hist(T)
```

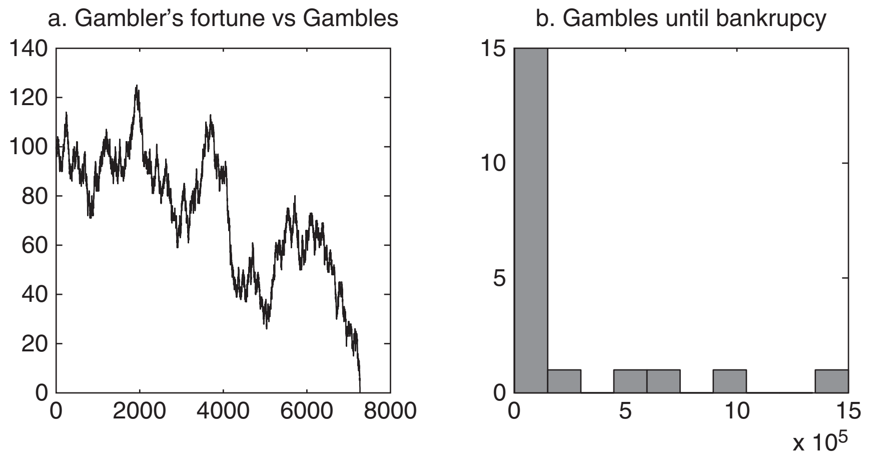


Fig. 1.5. a. Sample path of a gambler's ruin experiment. b. Histogram of the number of plays until bankruptcy in 20 trials.

1.2 Some Basic Probability

1.2.1 Events and Random Variables

By an *event* E in a probabilistic experiment, we mean some designated set of outcomes. The number of outcomes, or cardinality, of E is denoted by $|E|$. The set Ω of all the possible outcomes of the experiment is the *universe*. The probability of E, written $\Pr(E)$, is, in spirit, the fraction of times E occurs in an infinitely long sequence of trials of the experiment. More exactly, $\Pr(E)$ is the limit of this fraction for a finite number of trials as the number of trials increases unboundedly. A *random variable* is a mapping of the outcomes of an experiment, that is, Ω, to the real numbers.

For example, let the probabilistic experiment be tossing a fair coin twice. The outcome of each toss is heads (H) or tails (T); so the outcomes of the experiment are $\Omega = \{(H,H),(H,T),(T,H),(T,T)\}$. Some possible events are "both results are heads," "at least one head occurs," and so on. One possible random variable could be the number of heads that occur in a trial of the experiment. Let X denote this random variable; then X can be only 0, 1, or 2. Because each outcome in Ω is equally likely, the probabilities for X amount to just counting successful outcomes for the corresponding event. Thus $\Pr(X = 0) = 1/4$ because only (T,T) gives this outcome, $\Pr(X = 1) = 1/2$ since two outcomes give this event, and $\Pr(X = 2) = 1/4$ since only (H,H) qualifies here.

Suppose the outcomes of E are divided into two subsets such that $E = E_1 \cup E_2$ and E_1 and E_2 have no outcomes in common, that is, E_1 and E_2 are *disjoint events*, or $E_1 \cap E_2 = \emptyset$. Then the probability of occurrence of some outcome of E is just the sum of the probability that some outcome of E_1 might occur plus the probability that some outcome of E_2 might occur. Therefore

$$\Pr(E_1 \cup E_2) = \Pr(E_1) + \Pr(E_2) \quad \text{if } E_1 \cap E_2 = \emptyset. \tag{1.1}$$

For two general (not necessarily disjoint) events E_1, E_2, we have

$$\Pr(E_1 \cup E_2) = \Pr(E_1) + \Pr(E_2) - \Pr(E_1 \cap E_2). \tag{1.2}$$

The intersection must be subtracted as shown here because those events are counted twice in the first two terms on the right-hand side. More formally, the equation follows by writing both E_1 and E_2 as the disjoint union $E_i = (E_i \setminus E_j) \cup (E_i \cap E_j)$.

1.2.2 Discrete and Continuous Random Variables

Random variables can be discrete or continuous (or mixed). A *discrete random variable* is one that can assume only finitely many or countably infinitely many outcomes (but only one at a time of course). The coin-tossing experiment above has only finitely many outcomes, in fact just four. Therefore the random variable for the number of heads is a discrete random variable.

As an example of an experiment with a countably infinite number of outcomes, let X be the number of tosses of a coin until the first head appears. In principle, it could take any number of coin flips for this to happen, so X is a discrete random variable but is not finitely valued.

A *continuous random variable* X is one for which $\Pr(X = x) = 0$ for every real value x. There are many phenomena that can be considered continuous, and it is advantageous to do so from the standpoint of mathematical simplification. These include the passage of time, most biological dimensions, distance, speeds, and so on. Even though measuring such a variable is necessarily limited to finite resolutions, the measurements are nonetheless taken to be examples of continuous random variables.

Let X be a random variable, either discrete or continuous. The *cumulative distribution function*, or cdf, of X is defined as

$$\text{cdf}(x) = \Pr(X \leq x).$$

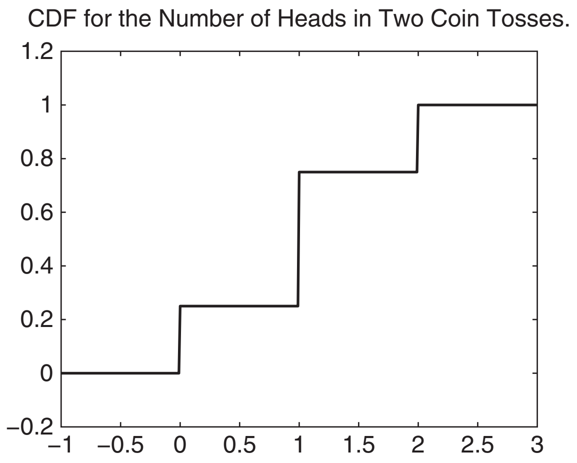


Fig. 1.6. Cdf for the number of heads in two coin tosses.

For example, consider the double coin-tossing experiment and random variable X from above. If $x < 0$ then $\mathrm{cdf}(x) = 0$, since there are no outcomes for which $X < 0$. At $x = 0$ the cdf jumps up to $1/4$, since $\Pr(X = 0) = 1/4$. Then at $x = 1$ the cdf again makes a jump, this time by $1/2$. Altogether $\mathrm{cdf}(1) = 3/4$, since the event $X \leq 1$ consists of the outcomes $\{(T, T), (H, T), (T, H)\}$; see Figure 1.6.

Between $x = 0$ and $x = 1$ the cdf is unchanged and so remains at the value $1/4$ over this interval. Similarly, between $x = 1$ and $x = 2$ the cdf has the constant value of $3/4$.

Finally, at $x = 2$ the cdf jumps by $1/4$ up to 1; $\mathrm{cdf}(2) = 1$ and remains there, since all possible outcomes are now included.

In general, as x tends to minus infinity, more and more outcomes of the experiment are omitted, and so the cdf must tend to 0,

$$\lim_{x \to -\infty} \mathrm{cdf}(x) = 0.$$

Similarly, as x tends to plus infinity, more and more outcomes are included, with the result that the cdf tends to 1,

$$\lim_{x \to \infty} \mathrm{cdf}(x) = 1.$$

Also it is true in general that as x increases the cdf either remains constant or increases as more and more outcomes satisfy the condition that $X \leq x$. That is, the cdf satisfies the condition that if $x_1 \leq x_2$ then $\mathrm{cdf}(x_1) \leq \mathrm{cdf}(x_2)$.

For a discrete random variable, the cdf will jump in value at the discrete points corresponding to those values of x for which one or more outcomes satisfy $X = x$. Otherwise, the cdf is constant. To illustrate this, Figure 1.7 shows the cdf for the experiment of rolling a pair of dice.

```
 Matlab
> red = floor(6*rand(1,1000))+1;% vector of 1000 random values 1,2,...,6
> green = floor(6*rand(1,1000))+1;% ditto for 2nd die
> dice = red+green;
> x=0:0.1:13; % vector of values from 0 to 13 incrementing by 0.1
> for i=1:length(x)
>  cdf(i)=sum(dice <= x(i));
> end
> plot(x,cdf)
```

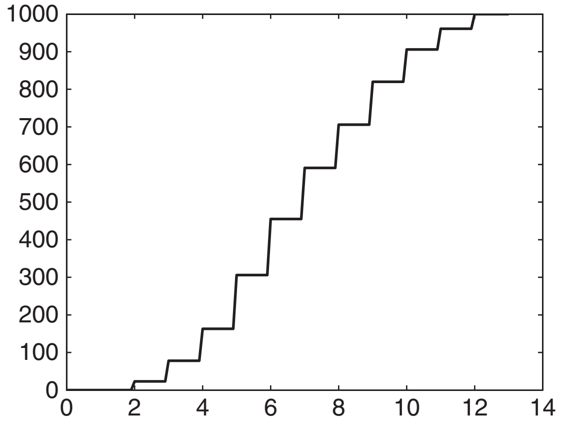


Fig. 1.7. Cdf for rolling a pair of dice.

For a continuous random variable the cdf either remains constant or increases continuously. For example, Figure 1.8 shows the cumulative distribution function for the radioactivity of a sample of carbon-14. The figure is based on the assumption that this material emits radioactive particles in any infinitesimally small amount of time with a probability proportional to the amount of carbon-14 remaining. Thus the figure shows the fraction of carbon-14 that has been converted to carbon-12 over the lifetime of the sample. The difference $1 - \text{cdf}(x)$ is the fraction of carbon-14 remaining. It is shown in the figure as the distance from the curve up to the horizontal asymptote cdf = 1. This measurement is used to date organic materials, since we know that the half-life of carbon-14 is about 5,700 years.

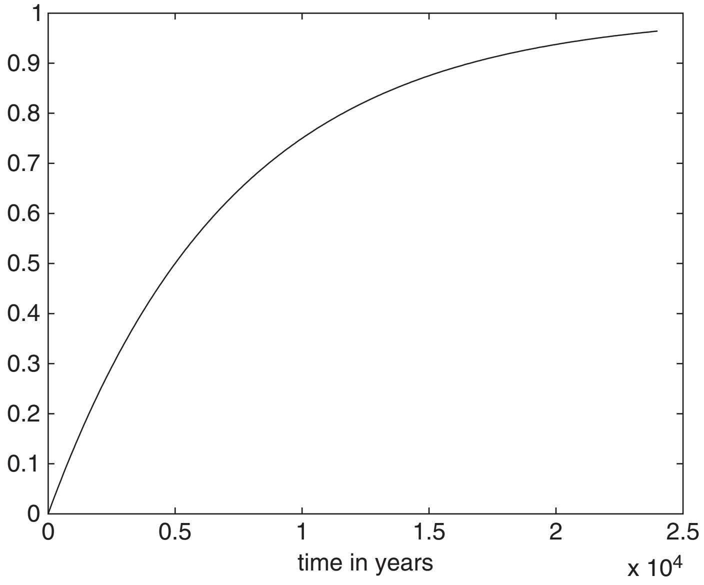


Fig. 1.8. Cdf for the amount of ^{14}C converted to ^{12}C.

1.2.3 The Probability Density Function

For a discrete random variable, the *probability density function*, or pdf, is the function $\mathrm{pdf}(x) = \Pr(X = x)$. Equivalently,

$$\mathrm{pdf}(x) = \lim_{\epsilon \to 0}(\mathrm{cdf}(x) - \mathrm{cdf}(x - \epsilon)).$$

Hence the probability that an outcome is (strictly) bigger than $x = a$ and less than or equal to $x = b$ is given by

$$\Pr(a < X \leq b) = \sum_{a < x \leq b} \mathrm{pdf}(x) = \mathrm{cdf}(b) - \mathrm{cdf}(a).$$

For a continuous random variable, the probability density function is the derivative of the cdf,

$$\text{pdf}(x) = \frac{d}{dx}\text{cdf}(x). \tag{1.3}$$

In this case the probability that an outcome lies between a and b is given by the integral

$$\Pr(a < X \le b) = \int_a^b \text{pdf}(x)dx = \text{cdf}(b) - \text{cdf}(a). \tag{1.4}$$

In particular, the cdf is given by

$$\text{cdf}(x) = \int_{-\infty}^x \text{pdf}(t)dt. \tag{1.5}$$

We illustrate this with the uniform distribution.

The Uniform Distribution

The most important distribution of Monte Carlo methods is the *uniform distribution*, because it is from this one that samples from all others are derived. The uniform distribution is a continuous distribution and gets its name because the value of the probability density function is the same for every point of Ω. That is, the pdf is constant on Ω.

One example is the uniformly distributed random variable X defined on the interval $\Omega = [0,6] = \{x : 0 \le x \le 6\}$. For every $x \in [0,6]$, the density is $1/6$, $\text{pdf}(x) = 1/6$, while $\text{pdf}(x) = 0$ for $x \notin [0,6]$. This is so because once again, $\Pr(\Omega) = 1$. Here

$$\text{cdf}(x) = \int_{-\infty}^x \text{pdf}(x)dt = \int_0^x \frac{1}{6}dt = \frac{x}{6}, \quad 0 \le x \le 6.$$

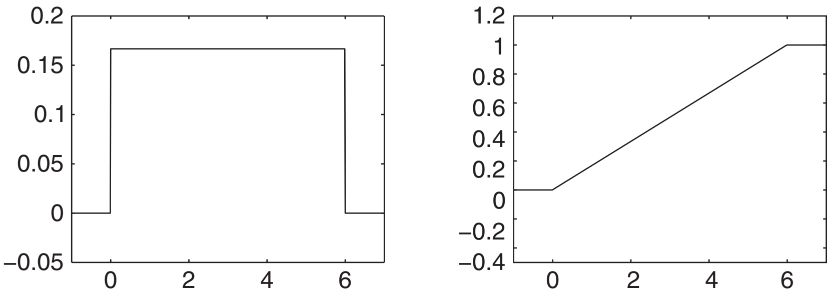
Then $\lim_{x\to\infty} \text{cdf}(x) = \text{cdf}(6) = 1$. This density and cumulative distribution are shown in Figures 1.9(a) and 1.9(b) respectively.

For convenience, denote by $U(a,b)$ the continuous uniform distribution on the interval $[a,b]$. Its density function is

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \le x \le b, \\ 0, & \text{otherwise}. \end{cases} \tag{1.6}$$

And the distribution function is

$$F(x) = \int_{-\infty}^x \frac{1}{b-a}dt = \int_a^x \frac{1}{b-a}dt = \frac{x-a}{b-a}. \tag{1.7}$$



(a) density (b) cumulative distribution

Fig. 1.9. Uniform distribution on $[0, 6]$.

We use the notation $U \sim U(0, 1)$ to mean a random variable U sampled from the distribution $U(0, 1)$; thus U is a uniform random sample from the unit interval $0 \leq U \leq 1$.[1] Note that any single real number x has 0 probability of selection from a continuous density because

$$\begin{aligned}\Pr(X = x) &= \lim_{\epsilon \to 0} \Pr(x - \epsilon < X \leq x) \\ &= \lim_{\epsilon \to 0} \int_{x-\epsilon}^{x} \mathrm{pdf}(t) dt = 0.\end{aligned}$$

Hence $U[a, b]$ and $U[a, b)$ and $U(a, b)$ are effectively all the same, and so we use the notation $U(a, b)$.

A (discrete) *equally likely* random variable is a random variable X defined on a finite set Ω and such that the pdf is constant. For example, if the cardinality of Ω is $|\Omega| = N$, then for every $x \in \Omega$, $\Pr(X = x) = 1/N$. This is so because the probability of *some* outcome occurring must be 1, that is, $\Pr(\Omega) = 1$. Said another way, at every value of the random variable X, the cdf takes a jump of $1/N$, and in the limit, $\lim_{x\to\infty} \mathrm{cdf}(x) = 1$.

1.2.4 Expected Values

Let r be a function defined on a random variable X, that is, on the outcomes of an experiment. If X is discrete then the *expected value*, or *expectation*, of r is the sum of its possible values $v = r(x)$ weighted by the probability of that value,

[1] More generally, if G is the name of some distribution, then $X \sim G$ will mean that X is a sample drawn from the distribution G.

$$\sum_{v=r(x)} v\Pr(X = x).$$

Alternatively, this can be written as

$$\mathbb{E}(r(X)) = \sum_{x\in\Omega} r(x)\,\mathrm{pdf}(x). \tag{1.8}$$

The expected value is denoted by $\mathbb{E}(r)$. Sometimes it is also called the mean or weighted average of r.

In the continuous case the sum becomes an integral, so this is

$$\mathbb{E}(r(X)) = \int_{-\infty}^{\infty} r(x)\,\mathrm{pdf}(x)dx. \tag{1.9}$$

The expectation value $\mathbb{E}(r)$ has a very simple interpretation. If an experiment is run many times, the expected value is the value we expect for the average of the values of the r's for all these trials of the experiment. That is, draw many independent *samples* from the random variable X, compute the values $r(X)$ for each of these samples, and average these values of $r(X)$; the value we expect for the average is $\mathbb{E}(r)$. This simple fact is the underlying basis for all Monte Carlo methods. The expected value is also useful as a measure of the "center" of the distribution of the values of r; that is, the histogram of these values tends to cluster around $\mathbb{E}(r)$.

For example, in the double coin-tossing experiment above, the expected number of heads in a trial of the experiment is given by

$$2\cdot\frac{1}{4} + 1\cdot\frac{1}{2} + 0\cdot\frac{1}{4} = 1,$$

since there is a one-fourth chance that two heads occur and a one-half chance that one head occurs. So in many runs of a double coin-tossing experiment, one expects on average 1 head per trial (two coins flipped).

Frequently occurring expectations are the *mean*, usually denoted by μ and defined as the expectation of the random variable itself,

$$\mu = \mathbb{E}(X) = \begin{cases} \sum_{\Omega} x\,\mathrm{pdf}(x) & \text{if } X \text{ is discrete,} \\ \int_{-\infty}^{\infty} x\,\mathrm{pdf}(x)dx & \text{if } X \text{ is continuous,} \end{cases} \tag{1.10}$$

and the *variance*, usually denoted by var and defined as the expectation of the square of the difference between the random variable and the mean,

$$\mathrm{var}(X) = \mathbb{E}((X-\mu)^2). \tag{1.11}$$

Note that $\mathrm{var}(X) \geq 0$, since $(X-\mu)^2 \geq 0$. An alternative formula for the variance is

$$\mathrm{var}(X) = \mathbb{E}(X^2) - \mu^2.$$

This is easily derived by noting that $(X-\mu)^2 = X^2 - 2\mu X + \mu^2$. Therefore

$$\begin{aligned}\mathbb{E}((X-\mu)^2) &= \sum_{x\in\Omega} \left(X^2 - 2\mu X + \mu^2\right) \mathrm{pdf}(x) \\ &= \sum_{x\in\Omega} X^2 \,\mathrm{pdf}(x) - 2\mu \sum_{x\in\Omega} X \,\mathrm{pdf}(x) + \mu^2 \sum_{x\in\Omega} \mathrm{pdf}(x) \\ &= \mathbb{E}(X^2) - 2\mu^2 + \mu^2 = \mathbb{E}(X^2) - \mu^2.\end{aligned}$$

The variance is a useful measure of the spread of a distribution about the mean. This is because the smaller the variance, the more the distribution is concentrated at the mean.

The derivation above highlights an important property of expectation, namely that it is *linear*. This is to say, if a and b are constants, and $f(X)$ and $g(X)$ are any functions of X, then

$$\mathbb{E}(af(X) + bg(X)) = a\,\mathbb{E}(f(X)) + b\,\mathbb{E}(g(X)). \tag{1.12}$$

One says that the expectation of a sum equals the sum of the individual expectations.

The variance is not, in general, linear. However, we do have

$$\mathrm{var}(aX) = a^2 \mathrm{var}(X) \tag{1.13}$$

for any constant value a.

As an example, we calculate the mean and variance of the roll of a pair of dice. We have

$$\begin{aligned}\mu &= 2\frac{1}{36} + 3\frac{2}{36} + 4\frac{3}{36} + 5\frac{4}{36} + 6\frac{5}{36} + 7\frac{6}{36} + 8\frac{5}{36} + 9\frac{4}{36} \\ &\quad + 10\frac{3}{36} + 11\frac{2}{36} + 12\frac{1}{36} \\ &= \frac{1}{36}\,(2+6+12+20+30+42+40+36+30+22+12) \\ &= \frac{126}{18} = 7.\end{aligned}$$

The variance is

$$\begin{aligned}\text{var} &= (2-7)^2\frac{1}{36} + (3-7)^2\frac{2}{36} + (4-7)^2\frac{3}{36} + (5-7)^2\frac{4}{36} + \\ &\quad \ldots + (10-7)^2\frac{3}{36} + (11-7)^2\frac{2}{36} + (12-7)^2\frac{1}{36} \\ &= 2\frac{25+32+27+16+5+0}{36} = \frac{105}{18} = 5\frac{5}{6}.\end{aligned}$$

As an example of this calculation for a continuous random variable, we calculate the mean and variance of the uniform distribution $U(a,b)$. Since the density of $U(a,b)$ is $f(x) = 1/(b-a)$, we have

$$\mu = \int_a^b x\frac{1}{b-a}dx = \frac{1}{b-a}\frac{x^2}{2}\bigg|_a^b = \frac{b^2-a^2}{2(b-a)} = \frac{b+a}{2}. \tag{1.14}$$

To get the variance, use the change of variable $y = x - (b+a)/2$,

$$\text{var} = \int_a^b \left(x - \frac{b+a}{2}\right)^2 \frac{1}{b-a}dx = \frac{1}{b-a}\int_{-\frac{b-a}{2}}^{\frac{b-a}{2}} y^2 dy.$$

Thus

$$\text{var} = \frac{1}{b-a}\frac{y^3}{3}\bigg|_{-\frac{b-a}{2}}^{\frac{b-a}{2}} = \frac{1/3}{b-a}\left[\left(\frac{b-a}{2}\right)^3 - \left(-\frac{b-a}{2}\right)^3\right] = \frac{1}{12}(b-a)^2. \tag{1.15}$$

Closely related to variance is the *standard deviation*, denoted by σ. It is defined as the square root of the variance,

$$\sigma(X) = \sqrt{\text{var}(X)}. \tag{1.16}$$

Like variance, the standard deviation is also a measure of the spread of a distribution. However, the standard deviation has the same units as X. That is, if X is measured in meters, $\text{var}(X)$ has as units meters^2, while $\sigma(X)$ has as units meters.

Expectations can be estimated empirically. If X_1, X_2, …, X_n is a sequence of samples from the distribution, then

$$\mathbb{E}(r) \approx \frac{1}{n}\sum_{i=1}^{n} r(X_i). \tag{1.17}$$

This is seen (in the discrete case) as follows. Let n trials be carried out and suppose n_1 outcomes were x_1, n_2 outcomes were x_2, and so on for the $k = |\Omega|$ possible outcomes. Then

$$\begin{aligned}\frac{1}{n}\sum_{i=1}^{n} r(X_i) &= r(x_1)\frac{n_1}{n} + r(x_2)\frac{n_2}{n} + \cdots + r(x_k)\frac{n_k}{n}\\ &\approx \sum_{x\in\Omega} r(x)\Pr(X=x).\end{aligned}$$

The last follows since $\Pr(X = x_i) \approx n_i/n$. Notice that this estimate of $\mathbb{E}(r)$ improves as n increases.

Chebyshev's Inequality

As mentioned above, the variance of a random variable X, $\mathrm{var}(X)$, is one measure of the spread of the distribution of X. *Chebyshev's inequality* is one precise way of saying this. Taking $a > 0$, the inequality says that

$$\Pr(|X-\mu| \ge a) \le \frac{\mathrm{var}(X)}{a^2}, \tag{1.18}$$

where $\mu = \mathbb{E}(X)$ is the mean of X.

In the case of simple distributions, like a uniform random variable $X \sim U(a, b)$, this inequality does not really give that much information. However, its real power comes in its generality: it applies to any distribution with finite mean and variance. For a proof see Figure 1.10.

Proof of Chebyshev's inequality for a discrete random variable X.
First assume that the random variable Z is greater than or equal to 0. Compute

$$\begin{aligned}\Pr(Z \ge a) &= \sum_{z\ge a}\Pr(Z=z) \le \sum_{z\ge a}\frac{z}{a}\Pr(Z=z) \le \frac{1}{a}\sum_{z} z\Pr(Z=z)\\ &= \frac{1}{a}\mathbb{E}(Z).\end{aligned}$$

Now notice that $Z = (X-\mu)^2 \ge 0$ and that $\mathbb{E}(Z) = \mathrm{var}(X)$.

Fig. 1.10. Proof of Chebyshev's inequality.

1.2.5 Conditional Probabilities

It often happens in a probabilistic experiment that additional knowledge makes some of the original outcomes impossible. For example, 5 cards are dealt from a randomized deck of 52 cards, but suppose the first 4 are known to be hearts. This information allows an improved calculation of the probability that all five cards will be hearts, a hand known as a *flush.*

In general, let A and B be events in Ω and suppose we want to calculate the probability of A given that the outcome must be restricted to B. This is referred to as the *conditional probability of A given B* and is written $\Pr(A|B)$. It may be calculated as

$$\Pr(A|B) = \frac{\Pr(A \text{ and } B)}{\Pr(B)}, \tag{1.19}$$

that is to say, B is acting as the universe here.

In the card deal example above (regarding order as important),

$$\Pr(5 \text{ cards out of 5 dealt are hearts}) = \frac{13}{52} \cdot \frac{12}{51} \cdot \frac{11}{50} \cdot \frac{10}{49} \cdot \frac{9}{48}$$

and

$$\Pr(\text{first 4 are hearts (any 5th card)}) = \frac{13 \cdot 12 \cdot 11 \cdot 10 \cdot 48}{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}.$$

Hence $\Pr(\text{heart flush}|\text{first 4 cards are hearts}) = 9/48$.

In another example, suppose we know that the sum of a dice roll is 8; what is the probability that one die was 3? The event B here is the set of 5 possible rolls; $\{6+2,\ 5+3,\ 4+4,\ 3+5,\ 2+6\}$. Since the event of rolling a 3 on one die and an 8 on the pair of dice consists of 2 of these rolls, we have

$$\Pr(A|B) = \frac{\Pr(A \text{ and } B)}{\Pr(B)} = \frac{\frac{2}{36}}{\frac{5}{36}} = \frac{2}{5}.$$

Were it not for the additional information, the probability that (at least) one die of dice roll is 3 would be 11/36 (seen by directly counting them among the 36 possible rolls, do not count the $3, 3$ outcome twice).

A very important use of conditioning is in the calculation of probabilities by *decomposition.* Suppose events B_1, B_2, …, B_n form a decomposition of the universe; that is, they have no events in common and together they include all the possible outcomes; symbolically,

$$\Omega = \bigcup_{i=1}^{n} B_i \quad \text{and} \quad B_i \cap B_j = \emptyset, \quad i \neq j.$$

Then for any event A, it follows that

$$\Pr(A) = \Pr(A|B_1)\Pr(B_1) + \Pr(A|B_2)\Pr(B_2) + \cdots + \Pr(A|B_n)\Pr(B_n). \tag{1.20}$$

Here is an example. AAA Auto Repair wants to know whether an alternator it ordered will arrive in two days. The parts house uses two delivery companies depending on availability. B-1 Delivery is available 30% of the time and has an 80% track record for two-day deliveries. Otherwise, hence 70% of the time, B-2 Package Service is used, but their two-day delivery record is only 40%. Let A denote the event the alternator arrives in two days, let B_1 be the event B-1 Delivery is used and B_2 the event B-2 Package Service is used. Then

$$\begin{aligned}\Pr(A) &= \Pr(A|B_1)\Pr(B_1) + \Pr(A|B_2)\Pr(B_2) \\ &= (0.8)(0.3) + (0.4)(0.7) = 0.52.\end{aligned}$$

Independent Events and Independent Random Variables

Two events are *independent* if the outcome of either one of them has no effect on the other. For example, in a coin toss of two coins, the outcome of one of the tosses has no effect on the other. If events A and B are independent, then

$$\Pr(A \text{ and } B) = \Pr(A)\Pr(B). \tag{1.21}$$

This is because $\Pr(A|B) = \Pr(A)$, since B has no effect on A and therefore

$$\Pr(A) = \Pr(A|B) = \frac{\Pr(A \text{ and } B)}{\Pr(B)},$$

from which the equation follows.

Two random variables X_1 and X_2 are independent if for any two subsets of real numbers $A, B \subset \mathbb{R}$, the events $E_1 = \{X_1 \in A\}$ and $E_2 = \{X_2 \in B\}$ are independent events. Again this means that knowledge of X_1 gives no knowledge about X_2.

Bayes's Formula and the Monty Hall Problem

Sometimes it is easy to confuse $\Pr(A|B)$ with $\Pr(B|A)$. For example, suppose an expert testifies in court that there is only a 1 in 2 million chance that the DNA evidence left at the crime scene will match the DNA of a random person from the population. Since the accused's DNA matched, the prosecution claims the probability that the accused is innocent is 1 in 2 million. In other words

$$\Pr(\text{innocent}|\text{DNA match}) = \frac{1}{2{,}000{,}000}.$$

But what the DNA evidence actually says is

$$\Pr(\text{DNA match}|\text{innocent}) = \frac{1}{2{,}000{,}000}.$$

So can the order of a conditional ever be reversed? Bayes's formula answers this question. Since $\Pr(A \text{ and } B)$ is symmetric in A and B, from (1.19) we can write

$$\Pr(A|B)\Pr(B) = \Pr(A \text{ and } B) = \Pr(B|A)\Pr(A).$$

Therefore one form of *Bayes's formula* is

$$\Pr(B|A) = \frac{\Pr(A|B)\Pr(B)}{\Pr(A)}. \tag{1.22}$$

Taking this one step further, the event A may be decomposed by the event B and its complement, which we write B^c; thus $\Pr(A) = \Pr(A|B)\Pr(B) + \Pr(A|B^c)\Pr(B^c)$. Hence *Bayes's formula* takes the form

$$\Pr(B|A) = \frac{\Pr(A|B)\Pr(B)}{\Pr(A|B)\Pr(B) + \Pr(A|B^c)\Pr(B^c)}. \tag{1.23}$$

Of course other decompositions of A are also possible, leading to other forms of this formula.

Returning to the DNA evidence problem above, if we want to reverse the conditional, we need the additional information as to an initial or a priori estimate of $\Pr(B)$, the innocence of the accused. Unfortunately this is not known. However, here is another situation, with a court backdrop, in which there is adequate information. This example is due to Kahneman and Tversky [KT73].

A hit-and-run incident after dark involves a taxicab. A witness says the cab was blue. In this town there are only two colors of taxis, 15% are blue and 85% are green. Under similar lighting and distance conditions it has been shown that a witness can make the correct color distinction between these two only 80% of the time. So what is the probability the cab was blue?

Since tests show that the color can be correctly identified 80% of the time, it would seem to be 80%. But this does not take into account the preponderance of green taxis. Using Bayes's formula we can do so. Let the event $A = \{\text{witness says the cab was blue}\}$ and event $B = \{\text{the cab was in fact blue}\}$. The alternative B^c is $G = \{\text{the cab was green}\}$. Then by Bayes,

$$\begin{aligned}\Pr(B|A) &= \frac{\Pr(A|B)\Pr(B)}{\Pr(A|B)\Pr(B)+\Pr(A|G)\Pr(G)}\\ &= \frac{(0.8)(0.15)}{(0.8)(0.15)+(0.2)(0.85)} = 0.4138,\end{aligned}$$

less than 50%.

As another simple application of Bayes's formula, we now look at the (in)famous *Monty Hall problem*, named after the television game show host of the same name. This particular problem has stirred a bit of controversy, primarily because there are subtle differences in the way the problem can be posed that lead to different solutions. The problem reads as follows [G92]:

A TV host shows you three numbered doors, one hiding a car (all three equally likely) and the other two hiding goats. You get to pick a door, winning whatever is behind it. You choose door #1, say. The host, who knows where the car is, then opens one of the other two doors to reveal a goat, and invites you to switch your choice if you so wish. Assume he opens door #3. Should you switch to door #2?

To obtain a mathematical description, let C_i be the event that the car is behind door i (for $i = 1, 2, 3$) and H_i be the event that the host opens door i. The question is asking for the conditional probability $P(C_2|H_3)$. By Bayes's formula this is

$$P(C_2|H_3) = \frac{P(H_3|C_2)P(C_2)}{P(H_3|C_1)P(C_1)+P(H_3|C_2)P(C_2)+P(H_3|C_3)P(C_3)}.$$

The crucial probability is $P(H_3|C_1)$. Questions 24 through 26 outline a more complete investigation of the Monty Hall problem.

1.2.6 Variance for a Sum of Random Variables– Joint Probability Densities

If X_1 and X_2 are two random variables with means μ_1 and μ_2, then

$$\begin{aligned}\operatorname{var}(X_1+X_2) &= \mathbb{E}((X_1+X_2-\mu_1-\mu_2)^2)\\ &= \mathbb{E}(\ [(X_1-\mu_1)+(X_2-\mu_2)]^2\ )\\ &= \mathbb{E}((X_1-\mu_1)^2)+\mathbb{E}((X_2-\mu_2)^2)\\ &\quad +2\mathbb{E}((X_1-\mu_1)(X_2-\mu_2)).\end{aligned}$$

In general, the term $\mathbb{E}((X_1-\mu_1)(X_2-\mu_2))$ does not have to equal zero. In fact, this term is called the *covariance* of the two random variables X_1 and X_2 and is a measure of how much they vary together. The covariance is denoted by $\mathrm{cov}(X_1, X_2)$,

$$\mathrm{cov}(X_1, X_2) = \mathbb{E}((X_1 - \mu_1)(X_2 - \mu_2)). \tag{1.24}$$

If X_1 and X_2 are independent, then $\mathrm{cov}(X_1, X_2) = 0$ and we get the following theorem.

Theorem 1.1. *If X_1 and X_2 are independent random variables, then*

$$\mathit{var}(X_1 + X_2) = \mathit{var}(X_1) + \mathit{var}(X_2). \tag{1.25}$$

Joint Probability Distributions

To see this, it is necessary to first discuss the *joint distribution* of the two random variables X_1 and X_2. If X_1 and X_2 are discrete, their *joint pdf* is the function

$$f_{X_1,X_2}(x_1, x_2) = \Pr(X_1 = x_1 \text{ and } X_2 = x_2).$$

This is a function defined on discrete points in the x_1, x_2 plane; these points make up Ω. As before, an event A is some subset of Ω, and

$$\Pr(X_1, X_2 \in A) = \sum_{(x_1,x_2)\in A} f_{X_1,X_2}(x_1, x_2).$$

We illustrate these constructions by two examples. First, a joint probability density can be made up from any two univariate densities simply by multiplying them together. Let X have density $f_X(x)$ and Y have density $f_Y(y)$; then

$$f_{XY}(x, y) = f_X(x) f_Y(y)$$

is a joint density. A concrete example of this is the joint pdf of rolling two dice, where

$$\Pr(\mathrm{Dice}_1 = i \text{ and } \mathrm{Dice}_2 = j) = \Pr(\mathrm{Dice}_1 = i)\ \Pr(\mathrm{Dice}_2 = j).$$

For the other example, let a probability experiment consist in selecting two cards from a deck of 8 cards made up of 2 clubs, 2 diamonds, 2 hearts, and 2 spades. Let X_1 be the first card drawn and X_2 the second, and let $f(x_1, x_2)$ denote the joint density. It is easy to see that

$$f(x_1, x_2) = \begin{cases} \frac{2}{8} \cdot \frac{2}{7} = \frac{1}{14}, & \text{if } x_1 \neq x_2, \\ \frac{2}{8} \cdot \frac{1}{7} = \frac{1}{28}, & \text{if } x_1 = x_2. \end{cases}$$

Getting a flush in this game is the event $A = \{(C,C),(D,D),(H,H),(S,S)\}$ and $\Pr(\text{flush}) = 4 \cdot \frac{1}{28} = \frac{1}{7}$.

For continuous random variables X_1 and X_2, the *joint cdf* is

$$F_{X_1,X_2}(x_1,x_2) = \Pr(X_1 < x_1, X_2 < x_2).$$

The pdf is the derivative of the cdf,

$$f_{X_1,X_2}(x_1,x_2) = \frac{\partial^2 F_{X_1,X_2}}{\partial x_1 \partial x_2}.$$

For a subset A of the x_1, x_2 plane,

$$\Pr(A) = \iint_{(x,y)\in A} f_{X_1,X_2}(x,y)\ dxdy.$$

Just as in the single-variate case, expectations are defined as the sum or integral over Ω. For example,

$$\mathbb{E}(r(X_1,X_2)) = \sum_{(x_1,x_2)\in\Omega} r(x_1,x_2) f_{X_1,X_2}(x_1,x_2).$$

Thus the covariance of (1.24) is defined in terms of the joint density of X_1 and X_2.

Marginal Distributions

Let $f(x_1,x_2)$ be a joint density function and define $f_1(x_1)$ and $f_2(x_2)$ by

$$f_1(x_1) = \sum_{x_2} f(x_1,x_2), \qquad f_2(x_2) = \sum_{x_1} f(x_1,x_2), \tag{1.26}$$

in the discrete case and

$$f_1(x_1) = \int_{-\infty}^{\infty} f(x_1,x_2)\ dx_2, \qquad f_2(x_2) = \int_{-\infty}^{\infty} f(x_1,x_2)\ dx_1, \tag{1.27}$$

in the continuous case. These are called the *marginals*. To get the marginal of the first variable, we sum or integrate out the second variable, and to get the marginal of the second variable, we do the same to the first variable. The marginals are univariate probability density functions and represent the distributions of each variable separately, regardless of the value of the other. That is, $f_1(x_1)$ represents the probability (density) of observing a given value of x_1 over all possible values of x_2.

It is easy to see that if the joint density is the product of two univariate densities as in our first example, $f_{X_1,X_2} = f_{X_1} f_{X_2}$, then f_{X_1} and f_{X_2} will be the marginals.

Continuing the eight card deck example above, the marginals are

$$f_1(x_1) = f(x_1, C) + f(x_1, D) + f(x_1, H) + f(x_1, S) = 3\frac{1}{14} + \frac{1}{28} = \frac{1}{4}$$

for any x_1. This says that the probability of any specific first card is one-fourth. Similarly, the x_2 marginal is

$$f_2(x_2) = f(C, x_2) + f(D, x_2) + f(H, x_2) + f(S, x_2) = 3\frac{1}{14} + \frac{1}{28} = \frac{1}{4}$$

for all x_2. Notice that $f(H,H) = 1/28$ but $f_1(H)f_2(H) = 1/16$. Therefore this joint distribution does not arise as the product of its marginals.

Now let f_{X_1,X_2} be the joint density of X_1 and X_2 and suppose these random variables are independent. It then follows that

$$\begin{aligned} f_{X_1,X_2}(x_1, x_2) &= \Pr(X_1 = x_1 \text{ and } X_2 = x_2) \\ &= \Pr(X_1 = x_1)\Pr(X_2 = x_2) = f_{X_1}(x_1) f_{X_2}(x_2). \end{aligned}$$

In other words, the joint density is made up of the product of two univariate densities just as in our first example.

Now finally we can see why the theorem about the sum of variances holds for independent random variables. In the discrete case we have

$$\begin{aligned} &\mathbb{E}((X_1 - \mu_1)(X_2 - \mu_2)) \\ &\qquad = \sum_{x_1}\sum_{x_2} (x_1 - \mu_1)(x_2 - \mu_2)\Pr(X_1 = x_1, X_2 = x_2) \\ &\qquad = \sum_{x_1}\sum_{x_2} (x_1 - \mu_1)\Pr(X_1 = x_1)(x_2 - \mu_2)\Pr(X_2 = x_2) \\ &\qquad = \left(\sum_{x_1} (x_1 - \mu_1)\Pr(X_1 = x_1)\right)\left(\sum_{x_2} (x_2 - \mu_2)\Pr(X_2 = x_2)\right) \\ &\qquad = \mathbb{E}((X_1 - \mu_1))\mathbb{E}((X_2 - \mu_2)). \end{aligned}$$

A similar argument works for continuous random variables.

The same definitions can be made for the joint distribution of more than two random variables, or even infinitely many random variables. In a similar way, the variance of a sum of independent random variables is always equal to the sum of the variances of the individual random variables.

1.3 Random Number Generation

In this section we take a first look at the problems and basic principles of random number generation. Since the computer is a deterministic device, it might seem impossible that it could be used to generate random numbers. This is correct: the numbers generated are algorithmically computed and are quite deterministic. However, they appear to be random and must pass stringent tests designed to ensure that they can provide the same results that truly random numbers would produce for the given problem. Consequently, computer-generated random numbers are sometimes termed "pseudorandom."

1.3.1 Requirements for a Random Number Generator (RNG)

The problem is to coax random-looking numbers out of a deterministic device. The desired qualities of a random number generator are these: It should

1. be fast–applications use millions of random numbers,
2. be repeatable–to make debugging possible,
3. be amenable to analysis–to ensure their distributional properties,
4. have a long period–all RNGs eventually repeat,
5. and be apparently random to the intended application–the quality of primary importance.

One class of techniques that have been tried are analog/digital hybrids, for instance, electronic circuits generating "white noise" or circuits listening for the clicks of a Geiger counter measuring radioactive decay. However, these methods have many drawbacks: they are too slow, not repeatable, tend to become biased, and require specialized equipment. Today, virtually all random number generators are algorithmic.

In Knuth's book *Seminumerical Algorithms, §3.1* (see [Knu81]), the author describes an early attempt on his part at constructing a random number generator. Despite being very complex, it nonetheless quickly converged to a fixed point! We describe Knuth's algorithm in order to illustrate that "... random numbers should not be generated with a method chosen at random."

The algorithm acts on 10-digit decimal numbers, changing the current number X into a new number in the sequence. The algorithm is given in the following steps:

1. Take N to be the most significant digit of X. Steps 2–13 are repeated exactly $N+1$ times.
2. Let M be the second most significant digit of X. Jump to step $3+M$.
3. If $X < 5 \times 10^9$, set $X = X + 5 \times 10^9$.
4. Replace X by $\lfloor X^2/10^5 \rfloor \bmod 10^{10}$. (The notation $\lfloor x \rfloor$ means the largest integer n with $n \leq x$.)
5. Replace X by $(1001001001 \times X) \bmod 10^{10}$.
6. If $X < 10^8$ then set X to be $X + 9814055677$; otherwise, set X to be $10^{10} - X$.
7. Interchange the lower-order five digits of X with the higher-order five digits of X.
8. Replace X by $(1001001001 \times X) \bmod 10^{10}$.
9. For each digit d of X, decrease d by 1 if $d > 0$.
10. If $X < 10^5$, set X to be $X^2 + 99999$; otherwise, set X to be $X - 99999$.
11. If $X < 10^9$, set X to be $10 \times X$ and repeat this step.
12. Replace X by the middle 10 digits of $X(X-1)$.
13. If $N > 0$, decrease N by one and return to step 2. If $N = 0$, the algorithm terminates with the current value of X as the next value in the sequence.

Unfortunately, the first time Knuth ran this algorithm it almost immediately converged to the value 6065038420 (which, amazingly enough, is converted to itself by this complicated algorithm). After this, when he ran it with a different starting value, it converged to a cycle having length 3178!

The lesson to be learned is that complexity is not a substitute for randomness, and in fact, sometimes apparent complexity hides simple behavior.

Generators Using One Stored Value

Ultimately a random number generator will be of the form

$$\text{output sample} = f(\text{variables}, \text{parameters})$$

for some function f. On each request for a random number, the algorithm calculates the prescribed function using the parameters and current values of its variables and outputs a “sample” from some hypothetical random process. As one request is made after another, the generator produces a sequence of samples, R_1, R_2, ... The parameters of the function are values fixed over the run of samples; what changes from sample to sample are the variables. Variables could be time or other quantities external to the algorithm, or one or more internally stored values that change from iteration to iteration. Internally stored variables are sometimes called *seeds*.

In this section we consider generators having only a single seed. If its value were assigned by some external process at any time during the production of samples, then it would be impossible to generate the same exact sequence of samples should the program need to be debugged. (Otherwise, the external process would be part of the function f.) It would also be hard to ensure the sequence's distributional properties. For this reason, the variable must be internally generated by the random number generator itself. An exception to this rule is in the assignment of the first value. Typically, a generator is "seeded" by the computer's clock. The initial seed should be saved by the application for debugging purposes if that becomes necessary.

As stated above, after its initial seeding, subsequent seeds are computed by the function f itself while it is simultaneously calculating the output sample. Matters are greatly simplified if the output sample and the next seed are the same thing! There is no loss in generality in this, since the application using the random numbers will adjust the samples to its own purpose anyway.

In any case, if the seed repeats at any time during the generation of the sequence R_1, R_2, $\ldots$, then from that point on, the sequence will repeat as well. Hence, as we said in the introduction, an RNG will repeat or cycle, and it is desirable to have as long a period as possible. Obviously, the period of a single-seed RNG can be no longer than the number of different possible seeds.

As a result of the foregoing, we see that a number generator is effectively like a book containing a long list of numbers. The initial seed selects the starting point for reading numbers from the list, and from then on, the numbers follow in predetermined order. Using a different starting seed simply begins reading from the list starting at a different place. If the same initial seed is used as on some previous occasion, then the sequence of "random" numbers generated is also the same. When one reaches the end of the list, the next number will come from back within the book somewhere (not necessarily from the beginning, see Problem 22 at the end of this chapter).

1.3.2 Middle-Square and Other Middle-Digit Techniques

The idea here is that after the first several digits in the calculation of most mathematical functions, such as sin() or log(), the digits are then essentially "random." To illustrate, the following code generates the digits of the sin() function after the fifth as the pseudorandom sequence. Generating a sequence of one hundred thousand of them might take half a second. This is a long time. By comparison, Matlab's built-in generator constructs the same number of samples in about 0.02 seconds. The reason is that computing the sin() function is a relatively time-consuming operation.

See Problem 17 for an investigation of the properties of this RNG.

```
 Matlab
> tenpow5 = 100000;
> x=0.5;
> for i=1:10
  x=tenpow5*sin(x);
  w = floor(x); % first 5 digits
  x = x-w % digits after the 5th
> end
>  % now do some timing
> tic
> for i=1:100000
  x=tenpow5*sin(x);
  w = floor(x);
  x = x-w;
> end
> toc %0.5314
>  % compare with Matlab's built-in generator
> tic;
> x=rand(1,100000);
> toc; %0.0193
```

In the early days of computing, calculating a transcendental function, such as sin(), was much more time-consuming than today, relative to the time for a multiplication for example. Hence in place of $\sin(x)$, von Neumann used simple multiplication, x^2, squaring the seed. Then, in the same spirit as above, he assumed that the middle digits of the result were random-like. Here is example code. A 7-digit integer is squared, resulting in a 13 or 14-digit result. The lowest 3 significant digits are dropped, then the next 7 digits are saved, and the remaining most significant digits are dropped. This then becomes the next seed.

```
  Matlab
> tenpow7 = 1000000;
> x = 0.1234567; % initial seed
> for i=1:10
   x = x*tenpow7; % x now an integer
   w = x*x; % square
   w = floor(w/1000); % drop last three digits
   w = w/tenpow7; % 7 decimal digits
   x = w - floor(w) % drop the integer digits
> end
```

Unfortunately, the middle-square generator suffers from several problems, one of which is a short period and rapid cycling for most initial seeds. The search was on for a better generator.

1.3.3 Linear Congruential Random Number Generators

The linear congruential random number generator is one of the most widely used but simple random number generators around. It depends on a simple recurrence to compute the next term in the sequence from the previous term. Before describing this generator, it is necessary to review the basics of modular arithmetic.

Modular Arithmetic

All the numbers in this section are integers. We use the notation $m \mid x$ to mean that m divides x (exactly, with no remainder). For example, $8 \mid 16$, the quotient being 2, or $8 \mid -24$ with quotient -3. One says that a is *congruent* to b modulo m, and writes

$$a \equiv b \bmod m, \qquad \text{if } m \mid (a-b).$$

For example $21 \equiv 5 \bmod 8$ since $8 \mid (21-5)$ or equally $-19 \equiv 5 \bmod 8$ since $8 \mid (-19-5)$.

Of course modulo m makes no sense if $m = 0$. And moreover, without loss of generality, we may, and do, take m positive in the rest of this section.

It is easy to see that congruence modulo m is:

reflexive: $a \equiv a \bmod m$,
symmetric: if $a \equiv b \bmod m$ then $b \equiv a \bmod m$, and
transitive: if $a \equiv b \bmod m$ and $b \equiv c \bmod m$, then $a \equiv c \bmod m$.

For transitivity, if $a \equiv b \bmod m$ then $a - b = pm$ for some integer p. Similarly, if $b \equiv c \bmod m$ then $b - c = qm$ for some integer q. Therefore $a - c = b + pm - b + qm = (p+q)m$, and so $m \mid (a-c)$.

Any relation that is reflexive, symmetric, and transitive supports equivalence classes. Fixing m, the *equivalence class* of the integer a, denoted by $[a]_m$ or just $[a]$ if m is understood, is the set

$$[a] = \{b : b \equiv a \bmod m\}.$$

For example, the equivalence class of 5 modulo 8 contains 5 itself, 21, -19 and an infinity of other integers. In fact, it contains all integers of the form $5 + 8p$, where p is any integer. Thus, to write all the numbers that are congruent to 5 modulo 8, just add or subtract multiples of 8:

$$\begin{array}{cccccccc} \cdots & -8 & -8 & -8 & & +8 & +8 & \cdots \\ \cdots & = -19 & = -11 & = -3 & = 5 & = 13 & = 21 & = \cdots \end{array}.$$

Note that the equivalence class of 21 modulo 8 is the same as the equivalence class of 5 modulo 8. In general, if two equivalence classes have any member in common, then they are in fact equal. Hence the equivalence classes are *mutually exclusive*, or disjoint. This is a consequence of transitivity. Therefore writing $[a] = [b]$ is the same as saying $a \equiv b \bmod m$.

An important fact about integers useful in working with modular arithmetic is the *division algorithm* (actually a theorem).

Theorem 1.2. *(Division Algorithm) Given two integers a and m, with $m > 0$, there exist unique integers q and r, the quotient and remainder, such that*

$$a = qm + r \quad \textit{and}\ 0 \le r < m.$$

The division algorithm shows that for any $m > 0$, the equivalence classes $[0]$, $[1]$, …, $[m-1]$ together contain all the integers, since any integer $a = qm + r$ is thus equivalent to r. Therefore the equivalence classes are *exhaustive* as well as exclusive.

In the following we will be interested in the member of the equivalence class given by the r of the theorem. To emphasize this, we use the notation $r = a \bmod m$ to mean $r \equiv a \bmod m$ and $0 \le r < m$.

Given m, an arithmetic can be defined for expressions of the form $(ax + by) \bmod m$. It is just ordinary arithmetic with a modulo m operation applied to the result. This is sometimes called *clock arithmetic*. Think of a clock whose dial is marked with the numbers $0, 1, \ldots, m-1$. It is easy to see that

$$(x \bmod m \pm y \bmod m) \bmod m = (x \pm y) \bmod m$$
$$((x \bmod m) \cdot (y \bmod m)) \bmod m = (x \cdot y) \bmod m.$$

For example, $(14 \bmod 11 + 31 \bmod 11) \bmod 11 = (3 + 9) \bmod 11 = 1$, but also $(14 + 31) \bmod 11 = 1$ These simple facts make implementing modular arithmetic quite easy.

Furthermore, if a is relatively prime to m (that is, a and m have no common factors other than 1), then we can even divide by a in arithmetic modulo m. That is, we can always solve equations of the form $ax \bmod m = b \bmod m$ for any b. This turns out to be quite useful and important.

The Linear Congruential RNG

We are now ready to define the *linear congruential random number generator* (LCRNG). Fix a, c, and m (the parameters) and take f as

$$f(x) = (ax + c) \bmod m, \qquad x \text{ an integer.}$$

For example, let $a = 5$, $c = 1$, and $m = 8$ and let the generator be seeded with $x = 0$. When called, the output sample generated is $(5 \times 0 + 1) \bmod 8 = 1$, and 1 becomes the next seed. Now continue; the sequence of samples will be

$$0, 1, 6, 7, 4, 5, 2, 3, 0,$$

whereupon the list cycles.

Of course, this sequence of numbers is not random, since it is generated by a rather simple formula; but with more realistic values of m and a, the values appear to be random.

It is immediately seen that the maximum possible number of different return values for a linear congruential generator is equal to m, the modulus. For this reason, in any real application, m is taken as large as possible, for instance 2^{32}. In fact, a choice for m of the form 2^k, where k is the wordlength of the computer, can be implemented very efficiently. This is because then, the modulo operation is free (takes no time and happens automatically, since there is no room for the higher-order bits). In this case, the LCRNG needs only one multiplication and one addition per result, and so is very fast. If floating-point numbers in the range 0 to 1 are desired, then a division by m must also be done for each number to be output.

By taking $c = 0$ even the addition can be avoided. In this case the sample sequence is given by $x_{n+1} = ax_n \bmod m$. Now the maximal period cannot be m, because $x_n = 0$ must be avoided, since then all subsequent samples would also be 0. But the period $m - 1$ can be achieved if m is a prime number. Of course in this case, m is not a power of 2, and so the modulo operation is no longer automatic on the binary computer as above. However, if one is willing to have a shorter period, it is still possible to get a period of $m/4$ and still use $m = 2^{\text{wordsize}}$ and thus still have automatic modulo operations.

More generally, when $c = 0$ (and m not prime), if x_n is a multiple of a divisor d of m then all subsequent values of x will also be multiples of d. For we will have $x_n = pd$ for some integer p, and if d divides m, then $m = qd$ for some integer q. So $x_{n+1} = ax_n \bmod m = apd \bmod m$ means $x_{n+1} - apd = rm$ for some integer r. But then $x_{n+1} = (ap+rq)d$ as stated. Therefore when $c = 0$, the x's should ideally run through all the integers relatively prime to m. With the right choice of a, it is possible for this to happen.

Notice that if 1 is in the sequence and $c = 0$, then the sequence is $1, a, a^2, \ldots, a^j, \ldots$ and thus will consist of all powers of a. If such a sequence comprises all nonzero equivalence classes modulo m (i.e. the entire set $\{1, 2, \ldots, m-1\}$), then a is said to be a *primitive element* modulo m.

Geometric View of the LCRNG

Why does the sequence of samples from the generator appear to be random? Have a glance at Figure 1.11, which shows f and the identity function $y = x$ (as continuous functions of the real variable x). At first f increases linearly with slope a and quickly reaches m. But then by the modulo operation, further increases are reduced by m, so the y values start again from 0. This continues, producing the sawtooth-shaped graph. Now the sequence of samples x_n is in reality a fixed-point iteration sequence

$$x_{n+1} = f(x_n)$$

on this graph. Starting with x_n on the x-axis, move vertically up to the sawtooth graph, this is $y = f(x_n)$. Now move horizontally at this value of y until you reach the identity function $y = x$, drop vertically down to the x-axis, and this is x_{n+1}. It is known that a fixed point iteration will converge to a fixed-point x^*, that is, a point satisfying $x^* = f(x^*)$, when the slope of the iteration function f is less than 1 in absolute value. But that is not the case here; the slope is $a > 1$. Far from converging, the iteration values bounce around trying to chase whichever sawtooth contains the current value of y. The fact that $a > 1$ makes the iteration have this "chaotic" behavior, which is what makes the LCRNG a good random number generator. Setting $a > 1$ also ensures that if you run the LCRNG twice with two slightly different initial seed values, even though the first few terms might be similar, the two sequences will quickly diverge.

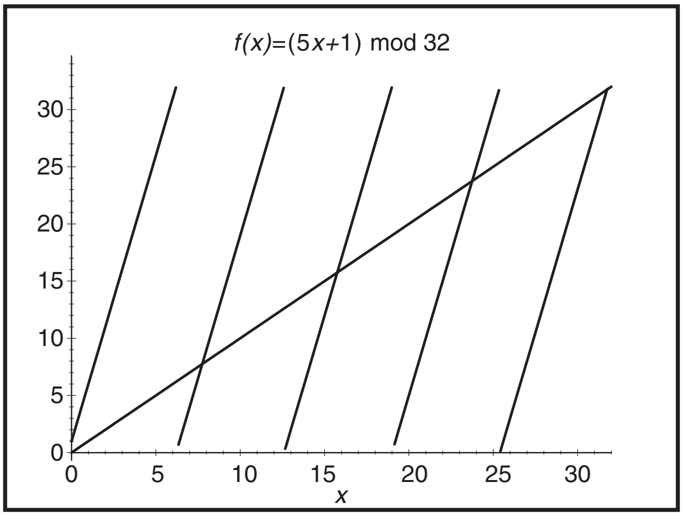


Fig. 1.11. Fixed-point iteration on a sawtooth.

Choice of Multiplier a

The following theorem shows how to obtain the maximal period of the LCRNG (see [BH62, Mar72]).

Theorem 1.3. *The linear congruential random number generator*

$$f(x) = (ax + c) \bmod m$$

has period m if and only if

1. *c is relatively prime to m (c and m have no common divisor other than 1) and*
2. *$a - 1$ is a multiple of p for every prime p dividing m, and*
3. *$a - 1$ is a multiple of 4 if m is a multiple of 4.*

As an example, for the modulus $m = 2^{32}$, the choices $c = 1$ and $a = 4\ell + 1$ work for any integer $\ell > 0$.

The Distribution of the Sample Sequence

Recall that we must be able to predict the distributional properties of the generator. This is easy to do for the single-seeded LCRNG. Over any period, the samples are all distinct, and at that point, they repeat. Hence the samples over the entire period are *uniformly distributed* over their range. Furthermore, because of the linear nature of the recurrence, they are also approximately uniformly distributed over segments of the period.

As we have seen, a generator engineered to have maximal period will likewise have as its range 0, 1, 2, …, $m-1$ (if $c \neq 0$). Hence these integers are sampled one after another, in "random" order, without repetition until all have been returned. Then the samples repeat in exact sequence. If the output values are divided by m, as is usually done, then the generator approximates uniformly distributed samples on $[0,1)$; approximates because only a discrete subset of m floating-point fractions are in fact returned. But then, m is often on the order of 2^{32}, so this is not so bad.

Improving the LCRNG

Many built-in pseudorandom number generators are linear congruential. This was especially so until recently. However, now better generators are available.

There is much room for improvement in the LCRNG. For one thing, the period is limited to at most $m-1$. Further, if the integers 0, 1, 2, …, $m-1$ were truly sampled randomly, repeats would be possible (although unlikely for large values of m), but this is *not* possible for linear congruential random number generators. And further analysis of the pseudorandom sequence turns up certain types of predictability, very undesirable for a good random number generator (see the exercises for examples of this). The single-seeded LCRNG is just too simple.

Improvements to the LCRNG include multivariate generators of the congruential type and generators using a different mathematical basis. We take up this topic again in Appendix A.

1.4 Some Applications

Hit-or-Miss

One of the first uses of the Monte Carlo method was in the evaluation of integrals. For example, let $\theta = \int_a^b f(x)dx$, the area under the function f between $x = a$ and $x = b$. We assume $f(x) \geq 0$ on $[a,b]$. A natural idea for estimating this area is *hit-or-miss*. By analogy, it is like throwing darts, blindfolded, at a target. Let $M \geq \max_{a \leq x \leq b} f(x)$; then the rectangle $[a, b] \times [0, M]$ encloses the target area (see Figure 1.12). Now proceed to select points at random in the rectangle. The probability that a point will fall within the target area is equal to the ratio of that area to the area of the rectangle; that is, the probability of a "hit" is $\theta/M(b-a)$. Therefore, if n, the number of points selected, is large, and h is the number of hits, then

$$\frac{h}{n} \approx \frac{\theta}{M(b-a)}.$$

In other words,

$$\theta \approx \left(\frac{h}{n}\right) M(b-a). \tag{1.28}$$

Note that this is exactly the technique used to estimate π in the Buffon needle problem.

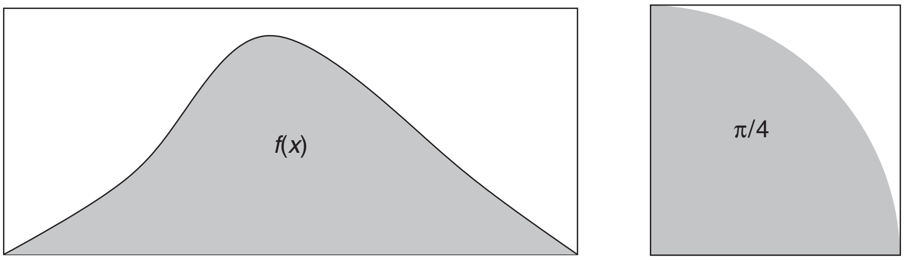


Fig. 1.12. Hit-or-miss integration.

The greatest use of Monte Carlo for evaluating integrals is in multiple integration over complicated regions in high-dimensional space. The same idea works there; the target region is enclosed in an easy-to-calculate "container," and points are selected at random in the container and tested to see whether they also hit the target region. The ratio of hits to throws equals the ratio of the target "volume" to that of the container.

Let us illustrate the idea by estimating π de novo. The curve $y = \sqrt{1-x^2}$, $0 \leq x \leq 1$, is the first-quadrant portion of a unit circle. The area under it is $\pi/4$. The target area is contained in the square $[0, 1] \times [0, 1]$, which has container area equal to 1. Hence by hit-or-miss, an estimate of π is given by

$$\pi \approx 4\frac{h}{n}.$$

The following code implements this strategy; it produced the estimate $\pi \approx 3.1316$ using $n = 10{,}000$ trials. As explained above, by dividing the 10,000 trials into batches one can obtain a normal distribution for the estimates in addition to the mean, and thus get an estimate of the error, which doesn't depend on a previous knowledge of the value of π.

```
 Matlab
> nTrials = 10000;
> x=rand(1,nTrials);
> y=rand(1,nTrials);
> hits=sum(x.^2 + y.^2 < 1)
> piEst = 4*hits/nTrials
```

Clearly the estimate of π improves as the number of trials, n, increases. In general, usually the best one can hope for is that the error decays at a rate bounded by $1/\sqrt{n}$. In Figure 1.13 we show the convergence of the estimate of π and the decay of the logarithm of the absolute error as we double n starting from $n = 2^2 = 4$ on the left to $n = 2^{20} = 1{,}048{,}576$. This figure is consistent with the error decaying at a rate of approximately $1/\sqrt{n}$, since the slope of a least-squares line would be somewhere around $-1/2$.

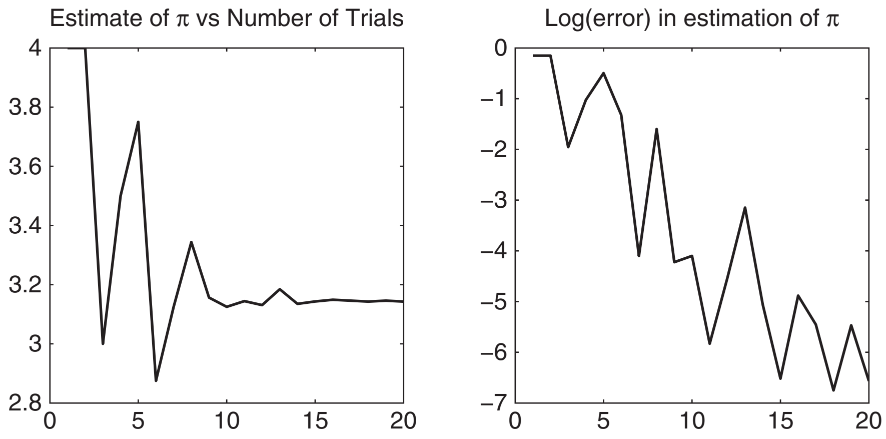


Fig. 1.13. (a) Convergence of estimate of π and (b) decay of absolute error.

Coupon Collecting

Retail companies often conduct a game in which the object is to collect all the letters in some word, for example their name. One of the letters is provided in each instance of their product. The question is, *How many products does one expect to have to purchase in order to collect all the letters?*

Usually in the commercial version of the game, one of the letters is rare; and the game degenerates into being lucky enough to get one of the instances having that letter. But in our game we will assume that each letter is equally likely to be acquired. With that, how many products will have to be purchased to spell, let us say, the nine-letter word *ANGELFISH*.

The following code will make the calculation. The results are shown in Figure 1.14. It shows that even though the letters occur equally likely, collecting all 9 letters could take 100 purchases or more. More interestingly, the number of products required peaks shortly after the minimum of 9 and then exponentially decreases.

```
 Matlab
> nLetters = 9; %ANGELFISH
> nTrials = 10000;
> for i=1:nTrials
>   success = 0;
>   nTries(i) = 0;
>   for j=1:nLetters
>     ANGELFISH(j)=0; %reset letter not achieved
>   end
>   while success == 0
>     nTries(i) = nTries(i)+1; %inc. count
>     buy = 1+floor(nLetters*rand); %letter obtained
>     ANGELFISH(buy) = 1;
>     if sum(ANGELFISH)==nLetters
>       success = 1;
>     end
>   end
> end
> hist(nTries)
```

For the coupon-collecting problem, it is possible to find an analytical solution. Thus, this is an example in which we can use Monte Carlo to make a probability calculation to suggest the way for an analytical solution. (See Problem 6 at the end of this chapter for another question related to the coupon collecting problem.)

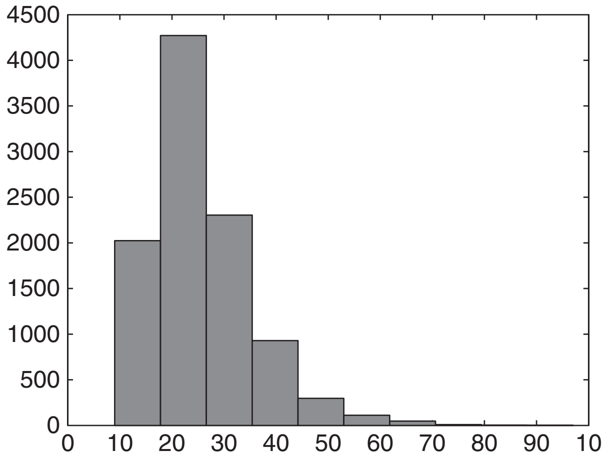


Fig. 1.14. Collecting the letters of ANGELFISH.

Shuffling Cards

In most card games the randomness in the game arises as a result of the initial randomization, or shuffling, of the cards. A shuffle may be viewed as a permutation of some fixed "home" order of the cards. The standard 52-card deck consists of 13 cards within a suit and 4 suits. Within a suit a canonical order could be ace, two, three, ..., ten, jack, queen, and king. Then the suits could be ordered, for example as in the card game bridge: clubs, diamonds, hearts, and spades. This identifies each card with a number from 1 to 52; in this case 1 is the ace of clubs, 14 the ace of diamonds, 27 the ace of hearts, and 40 the ace of spades. Generating a permutation of the integers 1 to 52 likewise produces a shuffle of the deck.

An investigation into the nature of a particular game would call for shuffles uniformly distributed over the space of permutations, that is, every permutation equally likely. Of course if one is after the nuances produced by human shuffling, then equally likely permutations are probably not desired and one may want to simulate the mechanical process of shuffling itself. See the problems about elementary shuffle operations at the end of Chapter 3.

Here we want to find an algorithm for generating equally likely permutations. To achieve this, it should be the case that having chosen some initial set of cards, the next card to be selected should be equally likely among the remaining cards. Of course the identity permutation, namely the order $1, 2, 3, \ldots, 52$, should also be generated with the same likelihood as any other permutation. And as always, the algorithm should be efficient, since thousands of shuffles may be required in an experiment.

Another desirable feature of a card shuffler is that it be able to generate only a partial shuffle if necessary. This saves time. For example, it may only be necessary to know the first m cards of the deck; then the cards are reshuffled.

The following algorithm fulfills all the criteria stated above. Further, it is accomplished within the original array; a second array is not needed. A card from the remaining deck is selected for position k as k runs from 1 to m. The selected card is exchanged with the card already in position k and selection continues.

```
 Matlab
> n = 4; % a permutation of 4 objects
> m = n; % do the whole deck
> perm = 1:n %identity permutation
> for i=1:m % if m=n the last is superfluous
>   j = i+floor((1+n-i)*rand); % from the remaining
>   swap = perm(i);
>   perm(i) = perm(j);
>   perm(j) = swap;
> end
> perm
```

Notes

The book [Ross05] is a classic textbook for basic probability theory. For a very enjoyable series of excursions in various topics in probability, see the book [ISAC95].

Appendix A has more information about generating random numbers and statistical tests of an RNG to ensure uniformity. Strong RNGs are absolutely necessary for cryptographic applications, and this has led to a renewed interest in both algorithmic and nonalgorithmic methods for generating random numbers. If "true" randomness is required, then hybrid methods involving physical sources of randomness are used.

Any algorithmic source of random numbers will have some structure, no matter how subtle, and hence there is likely to be some application for which this structure will be a problem. Thus it is prudent to try several different RNGs on a problem and compare the results. If there is some version of the problem for which the answer is known, this version can be used as a test case for the RNG. On the other hand, for some applications truly random numbers are worse than a sequence with some special type of structure. "Quasi-Monte Carlo" methods exploit this fact.

The "frequentist" interpretation of $\Pr(A)$ (for some event A) is as the percentage of the time that an experiment would yield a result in the set A. This interpretation (or definition, if you will) of probabilities then immediately leads to Monte Carlo methods of estimating probabilities, and hence expectations. There is another and very popular view, the *Bayesian view*, which interprets probabilities as some measure of your knowledge of the system. This is not just a philosophical distinction, however, since the Bayesian view often leads to different approaches to a problem. The advent of powerful computers has spurred the growth of Monte Carlo for Bayesian statistical methods, in particular Markov chain Monte Carlo methods (see [GH05]).

Problems: Chapter 1

1. (3) Perform a large number of trials of Buffon's needle problem and estimate the value of π. Plot the error as a function of the number of trials.

2. (4) Perform 2,000,000 trials of Buffon's needle problem for $1/\pi$: (a) by partitioning the trials into 10,000 "super experiments" of 200 trials each; (b) by partitioning the trials into 1000 "super experiments" of 2000 trials each; (c) by partitioning the trials into 100 "super experiments" of 20,000 trials each. Compare the results you obtain and comment.

3. (4) Simulate 1000 rolls of a pair of dice and histogram the results as (a) the sum of the dice and (b) the individual pairs, e.g. $(1,1)$ and $(1,2)$ and $\ldots$ through $(6,6)$. (c) Do the part (a) for the roll of 6 dice; histogram 4000 rolls.

4. (3) Suppose a "reward" $r(x)$ is given upon the roll of a pair of dice as follows: $r(x) = 1/x$, $x = 2, 3, \ldots, 12$. Estimate the expectation of r by simulation
$(\frac{1}{36}\left[\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \frac{5}{6} + \frac{6}{7} + \frac{5}{8} + \frac{4}{9} + \frac{3}{10} + \frac{2}{11} + \frac{1}{12}\right] = 0.1678)$.

5. (3) (Coupon collecting) (a) In boxes of cereal the manufacturer puts coupons bearing one of the letters of the word "milk." If all four letters are collected, the maker will send a coupon for a free carton of milk. Assume that the letters are distributed uniformly at random to cereal boxes. Simulate buying cereal boxes until all four letters have been assembled;

histogram the number of cereal boxes bought to achieve this. (b) Same question but now assume that "k" has 1/10 the probability of the others of occurring (i.e., Pr("m") = Pr("i") = Pr("l") = 10/31, Pr("k") = 1/31). (c) Same question as (a) but now collect the letters of "BattleCreek."

6. (4) (Modified coupon collectors) Generate a sequence of characters from the alphabet $\{a, b, c, d\}$ with each new character chosen with equal probability and independently of the previous character chosen. Now, slide a "window" of length 4 along this string and record the words that you get. How many characters do you expect to have to generate before you will have seen all possible $4^4 = 256$ different words? Use a simulation to answer this question. Run the simulation for alphabet sizes of $5, 6,$ and 7. Use "window" lengths of 4 and 5. Compare the results with those of the coupon-collectors problem.

7. (5) Simulate 1000 coin tosses. Record (a) the histogram, (b) the histogram for nonoverlapping pairs (x_n, x_{n+1}) then (x_{n+2}, x_{n+3}), etc., (c) the histogram for triplets (x_n, x_{n+1}, x_{n+2}), and (d) the histogram of runs (streaks), that is, a run of k for "heads" is k occurrences of heads in succession (and the next trial is "tails").

8. (5) (Random walk with drift) Use a biased coin to simulate a random walk of 30 steps on the line. If the coin falls heads (H), take one step to the right, if it lands tails (T), take one step left. After 30 steps, note the final position. Take $\Pr(H) = 0.6$ and $\Pr(T) = 0.4$. (a) Plot a sample path. (b) Make a histogram for 200 such random walks. (c) Report the sample mean. (d) Report the sample variance. (What should these be exactly?)

9. (4) (Gambler's ruin with time limits) Modify the simulation code in Section 1.1.3 starting on page 8 for the gambler's ruin problem, but with various upper bounds on the iteration count. How does this bias the average duration of the game? How does this bias the sample variance?

10. (4) By simulation, estimate the chance of being dealt a "pair" in poker. (Exact: $\left(13\binom{4}{2}\left(12\binom{4}{1}11\binom{4}{1}10\binom{4}{1}/3!\right)\right)/\binom{52}{5} = 0.422$).

11. (4) Same question as Problem 10 but instead for "two pair."

12. (5) Simulate the game of "craps." To start, a pair of dice are rolled. If the total is 7 or 11 on the first roll, then record a WIN and stop. If the total

is a 2, 3, or 12 on the first roll, then record a LOSE and stop. Otherwise, the total becomes the POINT (fixed for the rest of the game). Continue rolling until either a 7 occurs, and record LOSE, or POINT occurs (again) and record a WIN. Histogram WIN and LOSE for 1000 trials of the game.

13. (4) (Bayesian probability) An early detection test for HIV is 98% accurate. In a certain community, 3% of the population has HIV. What is the probability that a person with a positive test for HIV actually has the disease? Simulate this by selecting a person at random from the community and deciding whether the person has HIV. Then administer the test and decide whether it was positive. Keep a counter for the times the test is positive and for the times the test is positive and the person has HIV.

14. (4) Use the built-in random number generator on your system and sample from it to do the following. (a) Find the time per random (floating-point) number generation; i.e., how long does it take to generate 1,000,000 random numbers? (Representative times are between 0.2 μsec and 10 μsec per number.) Plot the following: (b) a sample path, i.e., if r_i is the ith random value generated, plot r_i vs i, (c) a cdf, (d) correlation plot, i.e., plot (x_n, x_{n+1}). Let $\delta = 1 \times 10^{-7}$; in 100,000,000 trials, how many times does the outcome lie in $[0, \delta)$, and how many times in $[1 - \delta, 1)$?

15. (4) Make up a linear congruential random number generator (LCRNG) and do the same as in Problem 14.

16. (4) Invent a non-LCRNG and do the same as in Problem 14.

17. Investigate the RNG using sin() given in Section 1.3.2. Use the same methodology as in Problem 14, above.

18. (4) (Difference between randomness and chaos) Answer the same questions as in Problem 14 for the random number generator $x_{n+1} = \lambda x_n(1 - x_n)$, $0 < \lambda < 4$, $0 \le x_i \le 1$. Take λ to be 3.95. Here, the x's are floating-point numbers, not integers.

19. (4) Answer the same questions as in Problem 14 but for the random number generator given by the sequences $x_{n+1} = 7 \times x_n \bmod 1$ (mod 1 means fractional part of). Again, the x's are floating point numbers, not integers.

20. (4) (Experiment with the multiplier a) Make up an LCRNG using, say, $m = 2^{16}$; sample it 100 times and find the length of the *runs-up* and the *runs-down*. For example, in the sequence

$$1,\ 6,\ 31,\ 156,\ 781,\ 3906,\ 847,\ 12,\ 67,\ 319,\ 1582$$

the first six numbers form a run-up, then comes a three-term run-down followed by a four-term run-up. Hence the runs-up total 10 in length and the runs-down total 3 in length. What should one expect from truly random numbers?

21. (4) Sample an RNG of your choosing and histogram the event $x_{n-1} < x_{n+1} < x_n$, i.e., histogram true/false or 0/1. What should be the probability of this event?

22. (3) For the LCRNG

$$x_{n+1} = 1 + 2x_n \bmod 2^{32}$$

show that there exist starting points x_0 that are never in a cycle (that is, the sequence generated by the LCRNG does not loop back to the starting point).

23. (3) Does the following algorithm produce equally likely permutations p of the integers $p_1 = 1$, $p_2 = 2$, …, $p_n = n$: for each i from 1 to n, let k be a randomly generated index, $1 \le k \le n$, swap p_k with p_i?

24. (3) (Monty Hall problem) The "standard" interpretation of the problem goes as follows: A car is randomly placed behind one of the three doors. The contestant randomly chooses one of the doors. If the contestant chooses the door with the car, the host randomly chooses one of the remaining doors to open and then gives the contestant a chance to either stick with the original choice or switch. On the other hand, if the contestant chooses a door with a goat, the host opens the other door with a goat and again gives the contestant a chance to either stick with the original choice or switch. Simulate this version of the game to obtain an estimate of the winning probability if the contestant sticks with the original choice and also if the contestant switches.

25. (3) (Monty Hall problem) Consider the following version of the Monty Hall problem. A car is randomly placed behind one of the three doors. The contestant chooses door #1. If the car is behind door #1, the host

chooses door #2 or #3 with equal probability to open. If the car is not behind door #1, then the host opens whichever of doors #2 or #3 that the car is NOT behind. Use simulations to estimate the probability of winning if the contestant switches or does not switch.

26. (3) (Monty Hall problem) The same as the previous problem, but if the car is behind door #1, the host chooses door #2 with probability q and chooses door #3 with probability $1 - q$.

27. (4) This exercise gives an example of a subtle problem that can occur using RNGs. Test the Matlab generator with the following program. To get the baseline result, first run the program with the two indicated lines commented out. Since the branch is executed one-tenth of the time, `low` should be about 1/10 of 10,000,000. Now restore the commented-out lines and rerun.

```
  Matlab
> low=0; high=0;
> for i=1:10000000
    x=rand;
    if x<0.1
      low = low + 1;
      y=rand; % for baseline comment this out
    else
      high = high + 1;
      y=rand; z=rand; %for baseline comment out
    end
  end
> low
> high
```

28. (5) (Monopoly) The following “Miniopoly” is a very rough approximation to the game of Monopoly. The round board has 40 “squares,” or landing places. Squares 0 and 40 are the same; and this square is called “Go.” A player starts with \$200 on Go, and on each turn, rolls a pair of dice to determine a move, which will be equal to the sum rolled. Squares 2, 7, 17, 22, 33, and 36 are “follow instructions on card” squares. The instructions are receive \$50, or \$100, or \$200 or pay \$100 or \$150 and are equally likely. Squares 0, 10, and 20 are “free”: nothing happens there. Square 30 is “Go to jail.” Passing or landing on Go pays \$200. Being in jail means that on one’s turn, if a roll is “doubles” (i.e., two 1’s or two 2’s or ... or two 6’s) then the player ”gets out of jail” and moves the value of the doubles from square 10; otherwise, the player pays \$10, stays in jail and doesn’t move. On any other square, like 29 for instance, pay

\$29 (to the "bank," an infinite source or sink of money). If at any time a player's money falls to \$0 or below, that player loses (and the game is over for that player). To be done: simulate a game of Miniopoly for 20 rolls, or bankruptcy, and observe the amount of money, the fortune, of the player at that time. Bankruptcy means fortune equal to \$0. Simulate many games and histogram the fortune.

29. (6) (Random probability distribution) The N points X_i are selected uniformly at random in $[0,1]$ and are then sorted so that $X_1 \le X_2 \le X_3 \le \cdots \le X_N$. Set $p_i = X_i - X_{i-1}$ for $i = 2, \dots, N$ and $p_1 = 1 + X_1 - X_N$. Notice that $p_i \ge 0$ and $p_1 + p_2 + \cdots + p_N = 1$. To start, set $N = 4$ and experimentally investigate both the individual distribution of the p_i's and the joint distribution of the vector $(p_1, p_2, \dots, p_N)$. Then you can try $N \ge 4$. Do all the p_i's have the same distribution (one way to test this is to do a histogram of each p_i separately and compare)? If so, what is this distribution? What about the distribution of the random vector? One way to explore the distribution of the random vector is to plot a *density plot*, as a surface, of two components at a time. A density plot here means that a region in $\mathbb{R}^2$ is partitioned into rectangular bins, counts are made of how many samples fall into each bin, and then a histogram is plotted as a surface.

    Another option goes as follows. For each sample $\mathbf{p} = (p_1, p_2, p_3, p_4)$, compute the point

    $$\mathbf{y} = p_1(1,0) + p_2(0,1) + p_3(-1,0) + p_4(0,-1) = (p_1 - p_3, p_2 - p_4)$$

    and do a density plot of these points. What should this plot look like if the distribution of $\mathbf{p}$ vectors is uniform? (It should be a simplicial spline function; see [BHR93].)

30. (5) (Random probability distribution) This problem tries another approach (other than that in Problem 29) to generate a set of probabilities on N points $(p_1, p_2, \dots, p_N)$ uniformly at random. The suggested method is to choose $X_i \sim U(0,1)$ for $i = 1, 2, \dots, N$ r and then set

    $$p_i = \frac{X_i}{\sum_i X_i}.$$

    Does this make all possible N-tuples $(p_1, p_2, \dots, p_N)$ have equal probability? Simulate this method and test for a uniform distribution. Plotting the distribution of each p_i separately should show that they all have the same distribution. Thus their joint distribution needs to be investigated. One possible method is to plot a joint pdf of two components at a time (like the density plot in Problem 29).

31. (3) (Batching needle tosses) What's wrong with batching the estimator #throws/#hits? Write a program to estimate π directly using just 100 throws. (For example, put throws = 100 in the program at the end of Section 1.1.1.) Replicate this experiment 2000 times and compute the average. (Hint: the exact value is $\sum_{i=1}^{100}(100/i)\cdot\binom{100}{i}p^i q^{100-i} = 3.21$ where $p = 1/\pi$ and $q = 1 - p$. See Section 2.3. The estimator #throws/#hits is *biased*, only approximating π as #throws $\to \infty$.)

2

Some Probability Distributions and Their Uses

The range of all possible probability experiments covers a lot of ground, from the outcome of a weighted coin toss to next month's price of a given stock, from the arrival time of the next phone call at a switchboard to the birth weight of a mother's first child. To simulate these and other probability experiments we need to know how to convert uniformly distributed random numbers into a sample from whatever distribution underlies the experiment.

In this chapter we introduce some of the major probability distributions and show how to use the computer to draw a random sample from each of them. This means converting samples U drawn from the uniform $[0, 1)$ distribution $U(0, 1)$ into samples from the required density. We will present the probability density and cumulative distribution functions for each distribution along with their means and variances. We will show how the random samples can be used to calculate probabilities and solve problems.

The main techniques used for sampling are *cdf inversion*, *simulation*, *composition*, *mapping*, and *rejection.* Our objective is to introduce these sampling methodologies and to illustrate them. In many cases, better, but more involved, methods are available for sampling these distributions more efficiently should timing be a critical issue. See for example [Rip87, BFS83, Rub81].

2.1 CDF Inversion–Discrete Case: Bernoulli Trials

Suppose an experiment has just two outcomes, say "failure" and "success" (or off and on, or 0 and 1, etc.). Such an experiment is called a *Bernoulli trial.* A coin flip is an example of a Bernoulli trial, and so is a roulette wheel with just two sectors. Let p be the probability of a success and $q = 1 - p$ the probability of failure. Then the pdf of a Bernoulli trial is (using 0 for failure and 1 for success)

$$\begin{aligned} f(x) &= \begin{cases} 1-p, & \text{if } x = 0, \\ p, & \text{if } x = 1, \end{cases} \\ &= p^x(1-p)^{1-x}, \qquad x = 0, 1. \end{aligned}$$

The cdf is

$$F(x) = \begin{cases} 0, & \text{if } x < 0, \\ 1-p, & \text{if } 0 \le x < 1, \\ 1, & \text{if } 1 \le x. \end{cases}$$

Consequently, using (1.10) and (1.11), the mean of a Bernoulli trial is

$$\mu = 0 \cdot (1-p) + 1 \cdot p = p$$

and the variance is

$$\text{var} = (0-p)^2(1-p) + (1-p)^2 p = p(1-p)\,(p + 1 - p) = pq.$$

2.1.1 Two-Outcome CDF Inversion

Drawing a Bernoulli trial sample T using $U \sim U(0,1)$ is very easy. If $U < p$ then declare that a "success" has occurred; otherwise, declare a "failure." This works because the $U(0,1)$ density is $\text{pdf}(t) = 1$ for $0 \le t \le 1$, and so

$$\Pr(T = \text{success}) = \Pr(U < p) = \int_0^p \text{pdf}(t)\, dt = p.$$

For example, suppose we want to simulate the spin of a casino roulette wheel and see whether the outcome is red. There are 18 red and 18 black sectors in a casino roulette wheel and two green sectors, 0 and 00. Therefore the success probability is $p = 18/38 = 0.4736\ldots$. Using the random number generator we generate U uniformly at random in $[0,1)$ and declare that "red" occurred if $U < 0.4736\ldots$; otherwise, "not red" occurred.

Alternatively, since $q = 1 - p = 0.5263\ldots$, we could test whether $U < q$. In this case our sample is "not red" if so and "red" if $U \ge 0.5263\ldots$. This is not a better method than before, but it is consistent with the cumulative distribution function that is shown in Figure 2.1. The interpretation is that U will lie between 0 and 1 on the y-axis. If U is less than $q = 1 - p$ return 0, meaning "not red" here, and if U equals or exceeds $1 - p$, return 1, meaning "red."

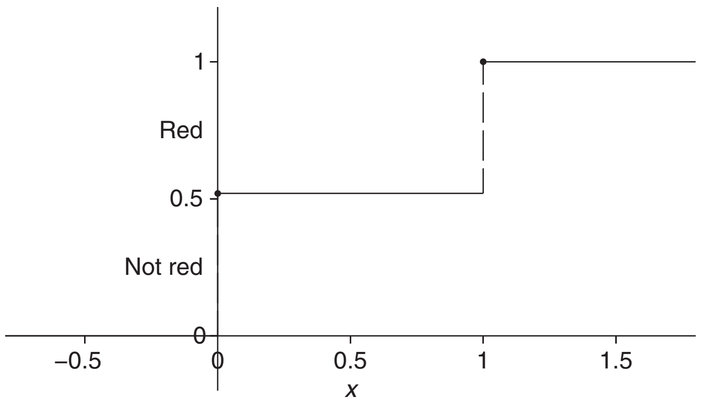


Fig. 2.1. Sampling for red.

2.1.2 Multiple-Outcome Distributions

The two-outcome Bernoulli trial is the simplest discrete distribution. More generally, discrete distributions have several possible outcomes. If their number is not too large, we can use the cdf graph to draw samples with the correct frequencies as hinted at above. Some examples will make this clear.

Suppose we want to simulate the number of eggs laid by a shorebird during breeding season. These birds have $X = 2$, 3, 4, or rarely 5 eggs per clutch. Suppose the observed frequencies of these sizes are $p_2 = 0.15$, $p_3 = 0.20$, $p_4 = 0.60$, and $p_5 = 0.05$ respectively. The cdf of X is shown in Figure 2.2.

Let $U \sim U(0,1)$ be a uniform $[0,1)$ sample. Starting from the point $(0,U)$ (on the y-axis), proceed to the right until you encounter a jump in the cdf, a vertical dashed line segment. Now proceed down to the x-axis, and return this value as the selection. As indicated along the y-axis, "2" is selected with 0.15 probability, since its interval occupies this fraction of the unit interval. Similarly, "3" is selected with 0.2 probability, since its interval occupies this fraction of the unit interval, and so on.

This technique is called *inverting the cdf*. For a discrete distribution it can be implemented on the computer in several ways.

The most general method sets up breakpoints in the interval $[0,1]$ and then determines into which subinterval $U \sim U(0,1)$ falls. The following code will generate one sample from X:

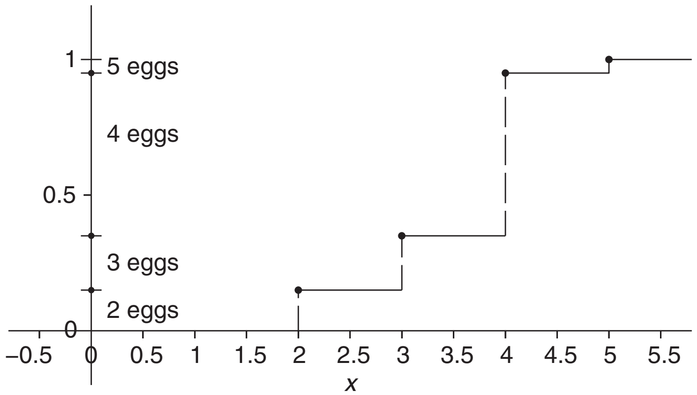


Fig. 2.2. Cdf for shorebird clutch size.

```
  Matlab
> % set up the outcomes
> oc(1)=2; oc(2)=3; oc(3)=4; oc(4)=5;
> % set up the breakpoints
> bp(1)=.15; bp(2)=bp(1)+.2; bp(3)=bp(2)+.60; bp(4)=1;
> % end of set up, go here to get samples
> U=rand; k=1;
> while( U>= bp(k))
   k=k+1;
  end % k is selected
> selected = oc(k)
```

A downside of this method is that the program must step through the `while` loop for each point returned. The search can be sped up by reordering the outcomes according to decreasing probability; this reduces the expected time in the `while` loop. For example, put the first breakpoint at 0.6 mapping to 4 eggs, then $0.6 + 0.2 = 0.8$ for 3 eggs and so on. In the first ordering, the expected number of times through the loop is

$$(1)(0.15) + (2)(0.2) + (3)(0.6) + (4)(0.05) = 2.55,$$

while in the second ordering the expected number of times through the loop is

$$(1)(0.6) + (2)(0.2) + (3)(0.15) + (4)(0.05) = 1.65.$$

While this may not seem like a great advantage, if thousands or millions of selections are needed it adds up.

As a much faster alternative, if the probabilities are simple two-place decimal fractions (as in this example), we can segment an array of length 100 into all the alternatives and then sample uniformly from this array. Here is the program.

```
  Matlab
> i=1;
> for j=1:15
   oc(i)=2; i=i+1;
  end;
> for j=1:20
   oc(i)=3; i=i+1;
  end;
> for j=1:60
   oc(i)=4; i=i+1;
  end;
> for j=1:5
   oc(i)=5; i=i+1;
  end;
>  % end of setup, go here to get samples
> k=floor(100*rand)+1;
> selected = oc(k)
```

Finally, the fastest way to generate a large number of samples is the vector method. The following code will generate 10,000 samples from $U(0,1)$.

```
  Matlab
> b(1)=.15; b(2)=b(1)+.2; b(3)=b(2)+.6;
> U=rand(1,10000); %10000 U(0,1) samples
> w2 = (U<b(1))*2;
> w3 = (U>=b(1) &U<b(2))*3;
> w4 = (U>=b(2) &U<b(3))*4;
> w5 = (U>=b(3))*5;
> w=w2+w3+w4+w5; % w a random 10000
>  % vector with the right frequencies
```

2.2 Walker's Alias Method: Roulette Wheel Selection

Multiple outcome distributions with arbitrary assignments of probabilities are like a (generalized) roulette wheel. It is no loss of generality to assume that the finite outcomes are labeled 1,2, ..., n. Assume that their corresponding probabilities are p_1, p_2, ..., p_n. Selecting an outcome is like spinning a roulette wheel with sectors having central angle $p_1 2\pi$ for outcome 1, $p_2 2\pi$ for outcome 2, ..., and $p_n 2\pi$ for outcome n.

The alias method takes more time to set up than cdf inversion, but runs faster because there is no search loop. The method uses two arrays, an array of probability adjustments $Q(\cdot)$ and an array of aliases $A(\cdot)$. Again we demonstrate the method by means of an example.

In some species the female can affect the fraction of males versus females among her offspring depending on various factors. For example, the rufous vanga female produces more male offspring if she mates with a large male but produces more female offspring if she mates with a small male (see [Asa05]). The rationale is that a large male will beget larger offspring, and large males mate more often and more widely than small males. So if she mates with a large male, she can more effectively propagate her genes by having males. Since females will always be able to mate, if she has small offspring, it is better to have females.

Suppose we want to sample the number X of female offspring in a clutch of size 8 given the statistics shown in Table 2.1. Since there are $M = 9$ possible outcomes, letting $Y = \text{int}[UM]$, the integer part of the product UM, where $U \sim U(0,1)$, selects $0, 1, 2, \ldots, 8$, equally likely, at about 11%. Therefore we use the Q array to adjust this down for those probabilities that need to be lower. Meanwhile, the excess probability is sent to some index $A(Y)$ whose selection probability exceeds 11%.

# female offspring	0	1	2	3	4	5	6	7	8
probability	0.02	0.04	0.08	0.15	0.2	0.21	0.16	0.09	0.05

Table 2.1. Probability for female offspring in a clutch of 8.

The method works like this.

Alias Algorithm

1. let $U \sim U(0,1)$ and put $Y = \text{int}[UM]$ and $Z = \text{frac}(UM)$, where $\text{frac}(x) = x - \text{int}[x]$ is the fractional part of x.
2. if $Q(Y)/M < Z$ return $X = Y$, else return $X = A(Y)$.

Arrays that work for our example problem are shown below:

Q	0.18	0.36	0.72	1.0	0.97	0.43	0.68	0.81	0.45
A	5	5	4	-	3	6	3	6	4

To see why this works, first note that Y is an equally likely random variable on $0, 1, 2, \ldots, M-1$, and Z is independent of Y and distributed as $U(0,1)$ (the argument that Y and Z are independent is that $[0,1)$ is partitioned into M equal parts, with Y selecting one of these parts and Z indicating *where* in this subinterval U is located). Therefore the **if** clause in Step 2 of the algorithm succeeds with probability $Q(Y)/M$ and selects Y. Otherwise, $A(Y)$ is selected with probability $(1-Q(Y))/M$.

Since the roulette wheel for $X=0$ is smallest, being 0.02, we start with it. Solving $Q(0)/M=0.02$ gives $Q(0)=0.18$, making index 0 occur with the correct frequency. We send the excess probability, amounting to $(1-Q(0))/M = 0.091$, to the index with maximum roulette wheel sector, namely 5, so $A(0)=5$. Now 9.1% of index 5's probability is taken care of, so it needs only $0.21-0.091=0.209$ more. We continue in this way using Q to adjust the uniform probability $1/M$ downward as needed and sending the excess probability to whatever index still needs more than $1/M=0.11$. If we send too much to some index, that causes no problem, since it too gets reduced by Q when that index's Q is calculated.

The alias algorithm is an ingenious implementation of the *composition* method of sampling. We have decomposed the original roulette wheel density f into two components: the first, f_1, is $Q(\cdot)/M$ with support on $0,1,\dots,8$ and the second, f_2, has support on the range of A, here $3,4,5,6$. The fraction of the time that Q decides the outcome is the sum $\sum_{i=0}^{8} Q(i)/9 = 0.622$, and the remaining fraction of the time, namely 0.377, the outcome is decided by A.

The composition method is a very general method and will be elaborated on shortly, in Section 2.8.

2.3 Probability Simulation: The Binomial Distribution

Often an entire series of independent Bernoulli trials is performed, and we are interested in the outcomes of all of them. Let the probability of a success be p and let $q = 1-p$. Let X stand for the number of successes in a series of n such trials. Then X is one of the values $\Omega = \{0,1,\dots,n\}$, but they are not equally likely. Even in our simple experiment of tossing two coins, recall that the probability of obtaining 1 head is 1/2, while the probability of 0 or 2 heads is 1/4 each. This is because the outcome 1 Head can occur in two different ways, (H, T) or (T, H).

The number of ways in which there can be x successes in n trials is given by the quotient

$$\frac{n(n-1)(n-2)\cdots(n-x+1)}{x(x-1)\cdots 2\cdot 1}, \tag{2.1}$$

in which there are x factors in both the numerator and denominator. This is seen as follows. Suppose first that the successes are distinguished from each other by subscripts, S_1, S_2, ..., S_x. Among the n trials there are n choices for where success S_1 occurs. To place S_2, one possibility has been taken (by S_1), so there are $n-1$ choices for where S_2 occurs. Continuing, we see that there are

$$n(n-1)\cdots(n-x+1) \tag{2.2}$$

ways of selecting on which trials the distinguished successes occur.

As an illustration of this, suppose the three symbols S, s, and F are used to make 4-letters words (we use upper- and lower-case S here instead of S_1 and S_2). The rule is that one and only one S must be used and the same for s. According to (2.2), there should be $4 \cdot 3 = 12$ such words. They are (S in first place) SsFF, SFsF, SFFs, (S in second place) sSFF, FSsF, FSFs, (S in third place) sFSF, FsSF, FFSs, and (S in fourth place) sFFS, FsFS, FFsS.

Returning to the general case, in reality the success are not distinguished, so if we pick one of the possible choices (of distinguished placements), how many other choices are merely rearrangements of the successes? But this is just counting the number of different orders of the subscripted S's, that is the *permutations* of x symbols. Using the same logic as above,

$$\begin{aligned}&\text{number of permutations of } x \text{ symbols} = \\ &\qquad\qquad x(x-1)(x-2)\cdots 2\cdot 1 = x!.\end{aligned} \tag{2.3}$$

As indicated, the notation for this product is $x!$, read x factorial. Hence the number of ways to obtain x successes in n trials is the quotient as predicted by (2.1). This ratio is denoted by $\binom{n}{x}$, read n *choose* x, or sometimes by $C(n,x)$,

$$C(n,x) = \binom{n}{x} = \frac{n(n-1)(n-2)\cdots(n-x+1)}{x(x-1)\cdots 2\cdot 1} = \frac{n!}{x!(n-x)!}. \tag{2.4}$$

It is easy to verify that the last member of this equation is an equivalent expression.

The quantity defined in equation (2.4) is also called a *binomial coefficient*, primarily because of its connection with the binomial theorem (given in equation (2.6)).

In our example, one of the words is SsFF, but there are $2! = 2$ ways of ordering S and s, so this word pairs off with sSFF. Likewise, SFsF pairs off with sFSF and so on. Hence, regarding S and s as not different, i.e., replacing s by S, there are six different possible words: SSFF, SFSF, SFFS, FSSF, FSFS, and FFSS. Similarly there are six different ways in which there can be two successes and two failures in four Bernoulli trials.

Now return to the problem of calculating the probability of x successes in n trials. We know that this can happen is $\binom{n}{x}$ ways; pick any one of them, for example all successes in the first x trials and all failures in the last $n-x$ trials. What is the probability of that? Since the trials are independent, it is the product $p^x q^{n-x}$ (recall that p is the probability of success and $q = 1-p$ is the probability of failure). This is also the probability of each of the other possibilities. Hence the probability that there are x successes is given by

$$\text{pdf}(x) = \Pr(X = x) = \binom{n}{x} p^x q^{n-x}, \qquad x = 0, 1, \ldots, n. \tag{2.5}$$

For the discrete random variable X, this is also its density function. This is the *binomial distribution*, often denoted by $b(n,p)$, n independent Bernoulli trials with probability p of success on each trial.

Equation (2.5) is closely related to the *binomial theorem* for the expansion of the binomial $(a+b)^n$,

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}. \tag{2.6}$$

In fact, summing (2.5) over x gives

$$\sum_{x=0}^{n} \binom{n}{x} p^x q^{n-x} = (p+q)^n = 1,$$

as expected, since $\Pr(\Omega) = 1$.

Mean and Variance

According to (1.10), the mean of the binomial distribution $b(n,p)$ is

$$\begin{aligned}\mu = \sum_{x=0}^{n} x \binom{n}{x} p^x q^{n-x} &= \sum_{x=1}^{n} \frac{n!}{(x-1)!(n-x)!} p^x q^{n-x} \\ &= np \sum_{x=1}^{n} \frac{(n-1)!}{(x-1)!(n-x)!} p^{x-1} q^{n-x}.\end{aligned}$$

Change variables by putting $y = x - 1$ and $m = n - 1$. Then $n - x = m - y$, and we have

$$\sum_{x=1}^{n} \frac{(n-1)!}{(x-1)!(n-x)!} p^{x-1} q^{n-x} = \sum_{y=0}^{m} \binom{m}{y} p^y q^{m-y} = (p+q)^m = 1.$$

Therefore

$$\mu = np. \tag{2.7}$$

As we have already seen, expectation is *linear*; recall (1.12). Since the binomial random variable is the sum of n independent Bernoulli trials, it follows that the mean of the binomial distribution is the n-fold sum of the mean of a Bernoulli trial, just as asserted by (2.7).

Likewise, by Theorem 1.1 of Section 1.2.6, the variance of the binomial random variable is the sum of the variances of the n independent Bernoulli trials,

$$\text{var} = npq. \tag{2.8}$$

2.3.1 Sampling from the Binomial

If n is "small," we could sample X using cdf inversion or the aliasing technique of the last section. Alternatively, and simpler to implement, a $b(n, p)$ sample may be generated by *simulation*, that is, by n repetitions of the underlying Bernoulli trial.

For example, suppose we are interested in how many times red occurs in 20 consecutive plays of the casino roulette wheel. Find out by repeating the Bernoulli trial 20 times: sample $U \sim U(0,1)$, check whether $U < 18/38$, and add 1 to the count if so.

If n is quite large, simulation is less desirable. Such is the case in the following problem. A mutation occurs to one of the daughter cells during cell division with probability $p = 10^{-7}$. In one million cell divisions of a laboratory culture, how many mutations occur? In the next section we will see how to do this without drawing one million Bernoulli samples.

2.4 Another Simulation: The Poisson Distribution

Consider a problem in which the number of Bernoulli trials is very large indeed. This is the problem of counting the number of "successes" of a phenomenon when the successes occur at a known rate λ. For example, suppose telephone calls come into a 911 emergency switchboard randomly, but on the average, 1 every 10 minutes. How many calls will arrive between 10 and 10:30 A.M.?

In each second of time there is a very slight probability that there will be an incoming call, so the arrival or nonarrival of an incoming call in any given second is a Bernoulli trial. Thus in 30 minutes there are 1800 such Bernoulli trials.

To analyze this, we make four assumptions about the events of the phenomenon: the event rate λ must be positive and

(a) The number of events occurring in nonoverlapping intervals are independent.
(b) The probability of exactly one event in a sufficiently short interval of length h is approximately λh.
(c) The probability of two or more events in a sufficiently short interval is negligible.

Let the random variable X_t denote the number of events that occur in an interval of length t. Partition the interval into n subintervals of equal length t/n, where n is large enough that conditions (b) and (c) hold. Then whether an event occurs in a given interval is a Bernoulli trial with success probability $\lambda t/n$, and therefore the probability of x events in the n subintervals is

$$\binom{n}{x}\left(\frac{\lambda t}{n}\right)^x\left(1-\frac{\lambda t}{n}\right)^{n-x}.$$

Let $n \to \infty$ in this expression. Group the $x!$ in the denominator of $\binom{n}{x}$ with $(\lambda t)^x$, since x is fixed. On the other hand, group $1/n^x$ with the remaining factors of the $\binom{n}{x}$ to get

$$\frac{n!}{n^x(n-x)!} = \frac{n(n-1)\cdots(n-x+1)}{n^x} \to_{n\to\infty} 1.$$

The remaining factor, $(1-\lambda t/n)^{n-x}$, tends to $e^{-\lambda t}$ as follows

$$\lim_{n\to\infty}\left(1-\frac{\lambda t}{n}\right)^{n-x} = \lim_{n\to\infty}\left(1-\frac{\lambda t}{n}\right)^{n}\left(1-\frac{\lambda t}{n}\right)^{-x} = e^{-\lambda t}1.$$

Therefore

$$\Pr(X_t = x) = \frac{(\lambda t)^x e^{-\lambda t}}{x!}. \qquad (2.9)$$

This is the density for the *Poisson* distribution. Note that λt is a pure (dimensionless) number. If the event rate is $\lambda = 1/10$ min and we want answers about a 30-minute period, then $\lambda t = 3$.

The Poisson density also applies to "events" distributed in space as well as time. For example, suppose that knots occur in a certain batch of lumber as a *Poisson process* with a rate of 2 knots per eight foot stud on average. What is the probability of finding a stud having no knots? By (2.9), since $\lambda t = (2/8) \times 8$ here, it is

$$\Pr(X_8 = 0) = \frac{(2)^0 e^{-2}}{0!} = e^{-2} = 0.135.$$

Note that 0! is 1 (by definition).

Since the mean of the binomial $b(n, \frac{\lambda t}{n})$ is $n\frac{\lambda t}{n} = \lambda t$ and is independent of n, it is natural to believe that the mean of the corresponding Poisson process is λt. This is indeed the case. Figure 2.3 gives the details.

Recall the mutation problem of the previous section. The Bernoulli trial probability p of a Poisson process is $\lambda t/n$, hence $\lambda t = np$. Therefore the Poisson approximation of this binomial is

$$\Pr(X = x) = \frac{(np)^x e^{-np}}{x!} = \frac{(0.1)^x e^{-0.1}}{x!}$$

for $n = 10^6$ and $p = 10^{-7}$. Compare this with (2.5). For $x = 0$ the exact probability is $(1-p)^n = 0.904837331$ to nine decimal places, while the approximation is $e^{-0.1} = 0.904837418$.

Although the Poisson distribution can be used as an approximation to the binomial, we still have the problem of drawing samples from it. As we will see next, the technique for sampling from a Poisson distribution exactly is by simulation. But both the Poisson and binomial distributions for large n can be sampled to a good approximation using the central limit theorem and the normal distribution covered below (see Section 2.6).

2.4.1 Sampling from the Poisson Distribution by Simulation

The following method will be explained in the exponential section to be taken up next. It is essentially a simulation technique, and is exact.

Let $U_0 = 1$ and U_i be $U(0,1)$ samples for $i = 1, 2, \ldots$. Return the greatest integer x such that

The calculation of the mean and variance of the Poisson distribution.

$$\begin{aligned}\mu = \mathbb{E}(X(t)) &= \sum_{k=0}^{\infty} k\frac{(\lambda t)^k}{k!}e^{-\lambda t}\\ &= e^{-\lambda t}\left[\frac{\lambda t}{0!} + \frac{(\lambda t)^2}{1!} + \frac{(\lambda t)^2}{2!} + \cdots\right]\\ &= \lambda t e^{-\lambda t}e^{\lambda t} = \lambda t.\end{aligned}$$

And the variance of a Poisson random variable is var $= \lambda t$:

$$\begin{aligned}\text{var} = \mathbb{E}((X(t)-\mu)^2) &= \sum_{k=0}^{\infty}(k-\lambda t)^2\frac{(\lambda t)^k}{k!}e^{-\lambda t}\\ &= e^{-\lambda t}\sum_0^\infty \left(k^2 - 2k\lambda t + (\lambda t)^2\right)\frac{(\lambda t)^k}{k!}\\ &= e^{-\lambda t}\sum_0^\infty \left(\frac{k^2}{k!}(\lambda t)^k - 2\frac{k}{k!}(\lambda t)^{k+1} + \frac{(\lambda t)^{k+2}}{k!}\right)\\ &= e^{-\lambda t}\Bigg[(\lambda t)\left(1 + \frac{2}{1!}(\lambda t) + \frac{3}{2!}(\lambda t)^2 + \cdots\right)\\ &\quad - 2\left(\frac{(\lambda t)^2}{0!} + \frac{(\lambda t)^3}{1!} + \cdots\right) + \left(\frac{(\lambda t)^2}{0!} + \frac{(\lambda t)^3}{1!} + \cdots\right)\Bigg]\\ &= e^{-\lambda t}\left[(\lambda t)\frac{d}{d(\lambda t)}\left(\lambda t + \frac{(\lambda t)^2}{1!} + \frac{(\lambda t)^3}{2!} + \cdots\right) - 2(\lambda t)^2 e^{\lambda t} + (\lambda t)^2 e^{\lambda t}\right]\\ &= (\lambda t)e^{-\lambda t}\frac{d}{d(\lambda t)}\left(\lambda t e^{\lambda t}\right) - (\lambda t)^2\\ &= (\lambda t)e^{-\lambda t}\left(e^{\lambda t} + \lambda t e^{\lambda t}\right) - (\lambda t)^2 = \lambda t.\end{aligned}$$

Fig. 2.3. Derivation of mean and variance of the Poisson distribution.

$$\prod_{i=0}^{x} U_i > e^{-\lambda t}.$$

This will be a sample from the Poisson distribution with parameter λ.

2.5 CDF Inversion, Continuous Case: The Exponential Distribution

The exponential distribution is used to model the random waiting time for "events" to occur. Examples of what we mean by events are customers arriving at a queue, the failure of mechanical components, and biological births or deaths. The assumptions about these events are the same as those made for the Poisson distribution. In fact, there is an intimate connection between the two.

The exponential random variable is the waiting time (or distance) for the first Poisson event to occur. Since the probability that a Poisson event will not occur in the interval $[0,t)$ is $e^{-\lambda t}$, as seen by taking $x=0$ events in (2.9), we have

$$\begin{aligned}\Pr(\text{an event occurs in } [0,t)) &= 1 - \Pr(\text{no event occurs in } [0,t)) \\ &= 1 - e^{-\lambda t}.\end{aligned} \tag{2.10}$$

This is the cdf of the exponential distribution,

$$\text{cdf}(t) = F(t) = 1 - e^{-\lambda t}.$$

The exponential distribution has a single parameter λ, the event rate; we denote the distribution by $E(\lambda)$. By its nature as the time at which an event occurs, the exponential is a continuous distribution. Therefore the exponential density is the derivative of its cdf,

$$\text{pdf}(t) = F'(t) = \lambda e^{-\lambda t}.$$

Likewise, probabilities are given by integrals. Let W be the random waiting time before an exponential event occurs. Then the probability that W occurs in some subset S of the real numbers is

$$\Pr(W \in S) = \int_S \lambda e^{-\lambda t}\, dt.$$

Properties of the Exponential Distribution

The exponential distribution is "memoryless" (also called *Markovian*). For example, if an event has not yet occurred by time a, then the probability that it occurs in the interval a to $a+b$ is the same as that of its occurring in 0 to b,

$$\Pr(W < a + b \,|\, W \ge a) = \Pr(W < b).$$

This is so because in terms of the cdf, the left-hand side is exactly

$$\frac{F(a+b)-F(a)}{1-F(a)} = \frac{(1-e^{-\lambda(a+b)})-(1-e^{-\lambda a})}{1-(1-e^{-\lambda a})}$$
$$= \frac{e^{-\lambda a}-e^{-\lambda a}e^{-\lambda b}}{e^{-\lambda a}} = 1-e^{-\lambda b}.$$

The mean of the exponential is $\mu = 1/\lambda$, because using integration by parts with $u = t$ and $dv = \lambda e^{-\lambda t}$, we have

$$\begin{aligned}\mu = \mathbb{E}(W) &= \int_0^\infty t\lambda e^{-\lambda t}dt = -te^{-\lambda t}\Big|_0^\infty - \int_0^\infty -e^{-\lambda t}dt \\ &= 0 + \frac{e^{-\lambda t}}{-\lambda}\Big|_0^\infty = \frac{1}{\lambda}.\end{aligned} \tag{2.11}$$

A similar calculation shows that the variance is

$$\text{var} = 1/\lambda^2; \tag{2.12}$$

see Figure 2.4.

Calculating the variance of the exponential distribution is done using integration by parts twice,

$$\begin{aligned}\text{var} = \mathbb{E}((W-\mu)^2) &= \int_0^\infty (t-\frac{1}{\lambda})^2\lambda e^{-\lambda t}dt \\ &= -(t-\frac{1}{\lambda})^2 e^{-\lambda t}\Big|_0^\infty + \int_0^\infty 2(t-\frac{1}{\lambda})e^{-\lambda t}dt \\ &= \frac{1}{\lambda^2} + 2\left[-\frac{1}{\lambda}(t-\frac{1}{\lambda})e^{-\lambda t}\Big|_0^\infty + \frac{1}{\lambda}\int_0^\infty e^{-\lambda t}dt\right] \\ &= \frac{1}{\lambda^2} + 2\left[-\frac{1}{\lambda^2} + \frac{-1}{\lambda^2}e^{-\lambda t}\Big|_0^\infty\right] = \frac{1}{\lambda^2}.\end{aligned}$$

Fig. 2.4. Derivation of the variance of the exponential distribution.

2.5.1 Inverting the CDF—The Canonical Method for the Exponential

The method for obtaining a sample from $E(\lambda)$ is simple. Let $U \sim U(0,1)$ and solve $U = 1 - e^{-\lambda W}$ for W. This will give

$$W = \frac{-1}{\lambda} \ln(1 - U),$$

and W is the desired sample. Note that most computer random number generators return values *strictly* less than 1, in which case the argument of the logarithm will not be 0. This is a general-purpose method, and its explanation is given by the following theorem.

Theorem 2.1. *(Inverting the cumulative distribution function, continuous case.) Let $F(x)$ be the cdf for the random variable X, suppose that F is continuous, and let $U = F(X)$. Then U is uniformly distributed on* $[0,1)$.

Proof. We need to show that $\Pr(U \le u) = u$ for $0 \le u < 1$. Since F is continuous and increasing, F^{-1} exists. If $x = F^{-1}(u)$ then

$$\begin{aligned}\Pr(U \le u) &= \Pr(F(X) \le u) = \Pr(X \le F^{-1}(u)) = \Pr(X \le x) = F(x)\\ &= u,\end{aligned}$$

as claimed. So if U is a sample from $U(0,1)$, then $X = F^{-1}(U)$ is a sample from cdf F.

Inverting the cdf also works for a distribution that is a mixture of discrete and continuous components.

Inverting the cumulative distribution, general case.
Sample $U \sim U(0,1)$; return

$$X = \inf F^{-1}([U,1]) = \inf\{\xi \,:\, F(\xi) \ge U\}.$$

For a proof of this, refer to Figure 2.5.

The main drawback to the method of inverting the cdf is that this usually requires an explicit formula for the cdf, which is often not available. However, in the cases in which it is available it is a very powerful technique.

If F is continuous, this becomes the same as in Theorem 2.1. Where F has jumps, note that F has to be right continuous, since

$$\lim_{\epsilon \to 0^+} \Pr(X \le x + \epsilon) = \Pr(X \le x).$$

Now assume for the moment the statement $X \le x$ if and only if $U \le F(x)$. Then we have

$$\Pr(X \le x) = \Pr(U \le F(x)) = F(x),$$

which was to be shown. As to the assumption, if $U \le F(x)$ then $x \in \{\xi : F(\xi) \ge U\}$, so $x \ge X$, which is the infimum of this set. On the other hand, if $X \le x$, then by monotonicity, $F(X) \le F(x)$, so $U \le F(x)$.

Fig. 2.5. Proof of the *inverting the cdf* technique in the general case.

Sampling from the Poisson Distribution

Since the Poisson random variable is the number of exponential events that occur in the interval $[0, t)$, we may simulate the Poisson distribution by sampling exponentials until time t is exceeded. We give here three methods to do this.

Method 1. Let W_i be $E(\lambda)$ and let K be the greatest integer such that $\sum_{i=1}^{K} W_i < t$.

Method 2. Let Y_i be $E(1)$ and let K be the greatest integer such that $\sum_{i=1}^{K} Y_i < \lambda t$. This works because if Y is $E(1)$, then $X = (1/\lambda)Y$ is $E(\lambda)$ (see below) and $\sum^K Y_i < \lambda t$ if and only if $\sum^K X_i < t$. This method is an improvement, since one does not have to divide by λ on each sample.

Method 3. (From Method 2 taking logs.) Let $U_0 = 1$ and $U_i \sim U(0,1)$ for $i = 1, 2, \ldots$. Let K be the greatest integer such that

$$\prod_{i=0}^{K} U_i > e^{-\lambda t}.$$

This works because

$$-\sum^{K} Y_i > -\lambda t \quad \text{if and only if} \quad \prod_{i=0}^{K} U_i > e^{-\lambda t},$$

and Y_i is $E(1)$ if and only if e^{-Y_i} is $U(0,1)$.

Multiple Independent Waiting Times

Suppose process A has waiting time W_A given by an exponential distribution with parameter λ_A, and process B has waiting time W_B given by an exponential distribution with parameter λ_B. The question is, *What is the distribution for the process consisting of the first of A or B to occur?* We assume that processes A and B are independent.

One can see that

$$\Pr(A \text{ or } B \text{ occurs in } [t, t + \varDelta t)|\text{not before}) \approx \lambda_A \varDelta t + \lambda_B \varDelta t,$$

with the approximation becoming exact as $\varDelta t \to 0$. Hence the event time for the combined process is exponential with parameter $\lambda_A + \lambda_B$.

Alternatively, we can argue from the cdfs thus:

$$\begin{aligned}\Pr(W_A \text{ or } W_B < t) &= \Pr(W_A < t) + \Pr(W_B < t) - \Pr(W_A \text{ and } W_B < t)\\ &= (1 - e^{-\lambda_A t}) + (1 - e^{-\lambda_B t}) - (1 - e^{-\lambda_A t})(1 - e^{-\lambda_B t})\\ &= 1 - e^{-(\lambda_A + \lambda_B)t},\end{aligned}$$

giving the same result.

2.5.2 Discrete Event Simulation

There are many situations in business and industry in which it is necessary to analyze processes that involve stochastic waiting times. This is the field of queuing theory. If one wishes to have an exact analysis of such a process, it is necessary for the waiting times to be exponentially distributed. This is so because the exponential is the only distribution that is "memoryless." That is, the future depends only on the present and not on how the present was arrived at.

Even so, anything beyond the simplest setup is too hard to analyze directly, so one turns to Monte Carlo simulation; it is known as *discrete event simulation* in this field. Among its many benefits, simulation allows any waiting time distribution to be used as easily as any other.

Due to its importance, there are several mature software solutions dedicated to providing answers for the most intricate of queuing situations. Much has been written about the area including the important mathematical problem of analyzing simulation results. In this section we give a necessarily brief description of one of the main approaches to discrete event simulation, that of *event scheduling.*

Event Scheduling Simulation

A software simulation is managed by a “simulation executive” that is responsible for stepping time through the simulation and invoking the other modules as necessary. These include modules for processing the entities of the simulation, implementing their interdependencies, and data display and logging modules.

A timetable of events is maintained, always in chronological order, and the simulation proceeds from one event to the next event in the schedule. Each event has a tag describing when it occurs in absolute time and its nature. When an event is handled, it may generate new events each of which is scheduled by sampling from the appropriate waiting time distribution and merging it into the master schedule at the appointed time.

Consider the example of simulating cell tissue growth, say starting with one cell at time 0 and carrying the simulation through for 8 hours. Assume that cell maturation times are exponentially distributed with parameter $\lambda = 0.5$ per hour. The master schedule needs to contain event times only, since all the events are the same, namely binary fission of identical cells. A more ambitious simulation could invoke crowding and other effects between the cells as desired. Drawing a sample from $E(0.5)$, we might get $W = 3$ hours, say. So our original cell divides at time $t = 3$ and our master schedule is $S[1] = 3$.

Now move time forward to $t = 3$ and handle the cell division of the first cell. Sample $E(0.5)$ again to get $W = 4$, say, for the first daughter cell, so now $S[2] = 7$ (the present time is $t = 3$ and the cell division time is 4, so the event is at absolute time 7). And again for the second daughter cell, get $W = 1$, say. Now we must shift the $S[2] = 7$ down to keep the schedule in time-sorted order, so $S[2] = 4$ and $S[3] = 7$.

The next event on the schedule is at time $t = 4$, the division of a first-generation daughter cell. Sample from $E(0.5)$ to get $W = 2$, say, and again to get $W = 5$. For the $W = 2$ event, its absolute time will be $4 + 2 = 6$, so the master schedule becomes $S[3] = 6$ and $S[4] = 7$.

The $W = 5$ event occurs at absolute time $t = 9$, which is beyond our study, so we need not include it in the master schedule. (But it would have to be accounted for if cell interactions were being monitored.)

Continuing in this fashion, we finally reach the point where all subsequent events occur beyond absolute time $t = 8$ and the simulation is over. At this point the statistics of the run are tabulated and presented.

2.5.3 Transforming Random Variables: The Cauchy Distribution

In the foregoing it was stated that if Y is $E(1)$, then $X = (1/\lambda)Y$ is $E(\lambda)$. Transforming or mapping random variables like this occurs frequently and is quite useful. In complete generality, a transformation of Y to X is given by $X = g(Y)$ for some one-to-one function g. To find the distribution of X (or to verify that it has some stated distribution) one can examine either the pdf or the cdf of X.

In our specific example we know that $\mathrm{cdf}_Y(y) = \Pr(Y \le y) = 1 - e^{-y}$, since Y is $E(1)$. Therefore we have the following chain of equalities:

$$\begin{aligned}\mathrm{cdf}_X(x) &= \Pr(X \le x) = \Pr\left(\frac{1}{\lambda}Y \le x\right) = \Pr(Y \le \lambda x)\\ &= 1 - e^{-(\lambda x)}.\end{aligned}$$

Thus the cdf of X is $1 - e^{-\lambda x}$, so X is $E(\lambda)$.

Alternatively, the pdf can be matched. In order to carry this out we have to recall the change of variables technique in integration. Let $t = g(\tau)$ and $A = [a, b]$ be an interval. Put $B = g^{-1}(A)$ (recall g is one-to-one). Then

$$\int_A f(t)\,dt = \int_B f(g(\tau))g'(\tau)\,d\tau. \tag{2.13}$$

The application to densities goes like this. With $X = g(Y)$,

$$\Pr(X \in A) = \Pr(g(Y) \in A) = \Pr(Y \in g^{-1}(A)) = \int_{g^{-1}(A)} f_Y(\tau)\,d\tau,$$

where f_Y is the pdf for Y. But by a change of variable,

$$\Pr(X \in A) = \int_A f_X(t)\,dt = \int_{g^{-1}(A)} f_X(g(\tau))g'(\tau)\,d\tau,$$

where f_X is the pdf of X. Equating the two end members yields

$$f_Y(\tau) = f_X(g(\tau))g'(\tau). \tag{2.14}$$

For our exponential mapping, $f_Y(\tau) = e^{-\tau}$, $g(\tau) = \tau/\lambda$, and, we anticipate, $f_X(\tau) = \lambda e^{-\lambda\tau}$. Substitution into (2.14) gives

$$e^{-\tau} = \lambda e^{-\lambda(\tau/\lambda)}\frac{1}{\lambda},$$

which is an identity, and thus we have verified the pdf of X.

Unfortunately, (2.14) is implicit for f_X. To get an explicit expression, we have to apply the change of variable to g^{-1} instead of to g. Doing so gives

$$f_X(x) = f_Y(g^{-1}(x)) \left(g^{-1}\right)'(x). \tag{2.15}$$

As a further example we work out the *Cauchy density* $f_X(t)$. Let $Y \sim U(-\pi/2, \pi/2)$ and $X = \tan(Y)$. Then we can compute that

$$f_X(t) = \frac{1}{\pi}\frac{1}{1+t^2}. \tag{2.16}$$

To see this, we notice that $g(t) = \tan(t)$, so that $g^{-1}(t) = \arctan(t)$ and thus

$$f_X(t) = \frac{1}{\pi}\frac{d}{dt}\left(\arctan(t)\right) = \frac{1}{\pi}\frac{1}{1+t^2},$$

as desired.

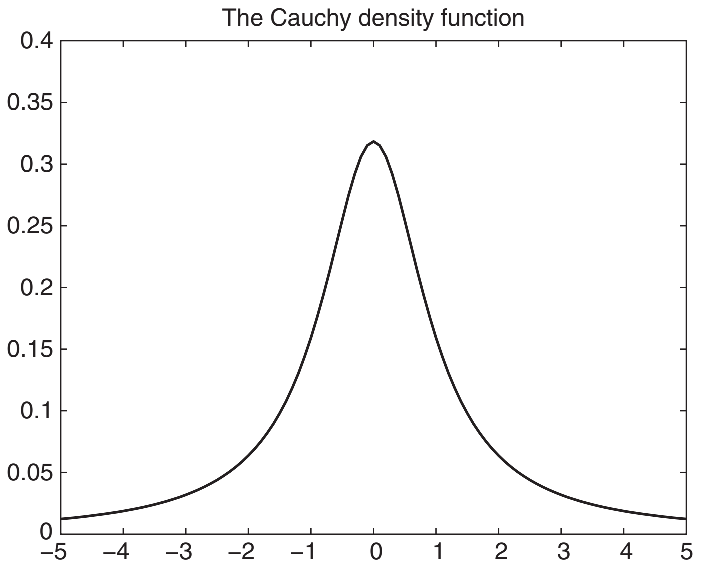


Fig. 2.6. The Cauchy probability density function.

The Cauchy density is symmetric about $t = 0$, and so its mean μ_X is 0; see Figure 2.6. But the variance of the Cauchy distribution is infinite, since the integral

$$\frac{1}{\pi}\int_{-\infty}^{\infty} \frac{t^2}{1+t^2} dt$$

is unbounded. Before Gauss's work, the Cauchy distribution was thought to be a candidate for the normal distribution, i.e., the distribution of the central limit theorem. Although the pdfs of the two distributions have a similar appearance, the tails of the Cauchy distribution are too "fat," as evidenced by the fact that the normal distribution has finite variance but the Cauchy distribution does not.

2.6 The Central Limit Theorem and the Normal Distribution

In many of the histograms generated by the problems we have worked on up to now, the figure that has emerged is that of a bell-shaped curve. This is not by accident; it is the consequence predicted by the central limit theorem, which we now consider.

The graph of the function $f(x) = e^{-x^2/2}$ (see Figure 2.7) is symmetric about the origin, everywhere positive, bell-shaped, and tends to 0 as $x \to \pm\infty$. Further, the integral of f over the whole real line is $\sqrt{2\pi}$,[1] and therefore

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

is a probability density function.

Replacing x by x/σ, where σ is a parameter, has the effect of rescaling the x-axis in terms of σ; that is, the rescaled function at $x = 2\sigma$ has the same value as the original function at $x = 2$. As a side effect, to maintain a unit integral, f must be divided by σ as well.

Finally, replacing x by $x - \mu$ has the effect of shifting the function; the shifted function at $x = \mu$ has the same value as the original at $x = 0$.

With these changes, the *normal* probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{2.17}$$

Denote this distribution by $N(\mu, \sigma^2)$. By the transformation principles for random variables (from the last section), it is easy to see that if X is distributed as $N(\mu, \sigma^2)$, then $Y = (X - \mu)/\sigma$ is distributed as $N(0, 1)$.

The mean and variance of the $N(\mu, \sigma^2)$ distribution are, appropriately enough, μ and σ^2 respectively. The calculation is given in Figure 2.8.

[1] This can be seen by calculating $\int_{-\infty}^{\infty} e^{-x^2/2} dx \int_{-\infty}^{\infty} e^{-y^2/2} dy$ in polar coordinates.

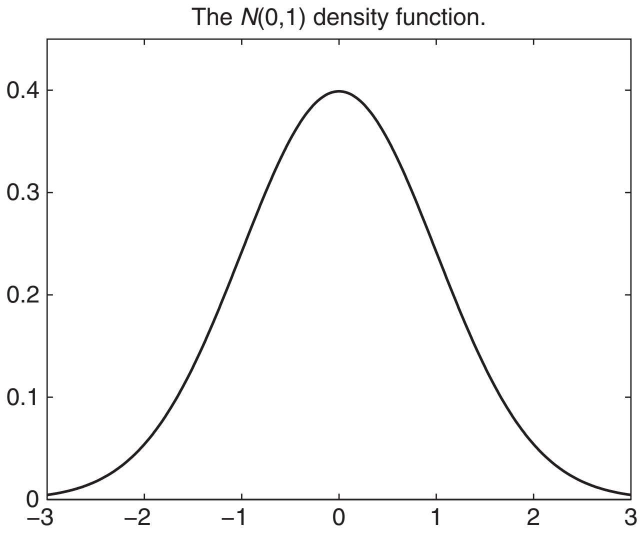


Fig. 2.7. The standard normal density function.

Obtaining the cumulative distribution function for the normal distribution is harder to come by. There is no antiderivative for e^{-x^2} in terms of elementary functions and so there is no simple form for the normal cumulative distribution function. However, approximate values may be computed by the technique of rational approximation, that is, the cdf of $N(0,1)$ is approximated by a rational function. Table 2.2 was computed using the code below, which uses this technique. Amazingly, the rational approximation produces values correct to three decimal places for all values of x, as in the table.

```
 Matlab
> function w = normalCDF(x)
> a1 = 0.31938153; a2 = -0.356563782;
> a3 = 1.781477937; a4 = -1.821255978;
> a5 = 1.330274429; b=0.2316410; c=1/sqrt(2*pi);
> L=abs(x); K=1./(1+b.*L);
> w = 1-c*exp(-L.*L/2).*...
>  (a1.*K+a2.*K.^2+a3.*K.^3+a4.*K.^4+a5.*K.^5);
> msign = -(x<0); mfact = 2*msign+1;
> w = mfact.*(w+msign);
```

Theorem 2.2. *(Central limit theorem) Let X_1, X_2, ..., X_n be independent random samples from a distribution with mean μ and finite variance σ^2. Then*

Let $u = (x - \mu)/\sigma$; then $x = \sigma u + \mu$ and $du = dx/\sigma$ and we have

$$\begin{aligned}
\mathbb{E}(X) &= \int_{-\infty}^{\infty} \frac{x}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} du = \int_{-\infty}^{\infty} \frac{\sigma u + \mu}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} du \\
&= \frac{\sigma}{\sqrt{2\pi}} \int_{-\infty}^{\infty} u e^{-\frac{1}{2}u^2} du + \mu \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} du \\
&= \mu,
\end{aligned}$$

since the first integral is 0 by symmetry and the second integral is 1 since f is a pdf. To calculate the variance we must add integration by parts to the substitution used above,

$$\begin{aligned}
\mathbb{E}(X^2) &= \int_{-\infty}^{\infty} \frac{x^2}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} du \\
&= \int_{-\infty}^{\infty} \frac{\sigma^2 u^2 + 2\sigma\mu u + \mu^2}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} du \\
&= \frac{\sigma^2}{\sqrt{2\pi}} \int_{-\infty}^{\infty} u^2 e^{-\frac{1}{2}u^2} du + \frac{2\sigma\mu}{\sqrt{2\pi}} \int_{-\infty}^{\infty} u e^{-\frac{1}{2}u^2} du \\
&+ \mu^2 \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} du.
\end{aligned}$$

The second member on the last line is zero by symmetry, and the third integrates to μ^2. For the first member, use integration by parts and get

$$\begin{aligned}
\frac{\sigma^2}{\sqrt{2\pi}} \int_{-\infty}^{\infty} u^2 e^{-\frac{1}{2}u^2} du &= \frac{\sigma^2}{\sqrt{2\pi}} \left[u e^{-\frac{1}{2}u^2} \Big|_{-\infty}^{\infty} + \int_{-\infty}^{\infty} e^{-\frac{u^2}{2}} du \right] \\
&= \frac{\sigma^2}{\sqrt{2\pi}} (0) + \sigma^2 \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{u^2}{2}} du \\
&= \sigma^2.
\end{aligned}$$

Therefore the variance is σ^2, since

$$E\left((X - \mu)^2\right) = \mathbb{E}(X^2) - \mu^2 = \sigma^2.$$

Fig. 2.8. Mean and variance calculation for the normal distribution.

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$F(x)$	0.500	0.540	0.579	0.618	0.655	0.691	0.726	0.758	0.788	0.816
x	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9
$F(x)$	0.841	0.864	0.885	0.903	0.919	0.933	0.945	0.955	0.964	0.971
x	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9
$F(x)$	0.977	0.982	0.986	0.989	0.992	0.994	0.995	0.997	0.997	0.998

Table 2.2. Computed approximate normal cdf.

$$Y = \frac{\sum_{i=1}^{n} X_i - n\mu}{\sqrt{n\sigma^2}}$$

has a limiting distribution as $n \to \infty$ *and it is* $N(0,1)$, *normal with mean* 0 *and variance* 1.

Exercise 29 gives empirical evidence for the central limit theorem.

2.6.1 Sampling from the Normal Distribution

We discuss three methods for sampling from the normal distribution. For a comparison of these methods, see Problem 26 at the end of this chapter.

Our first method takes advantage of the central limit theorem and is only crudely approximate. Its advantage is that it is quick and easy to program.

Central Limit Algorithm

Sample n copies of $U(0,1)$ random variables, U_1, U_2, …, U_n, and return

$$X = \frac{\sum_{i=1}^{n} U_i - \frac{n}{2}}{\sqrt{n/12}}.$$

By the central limit theorem, X is approximately normal with parameters $\mu = 0$ and $\sigma = 1$, that is, $N(0,1)$. Recall from equations (1.14) and (1.15) that the mean and variance of a $U(0,1)$ random variable are 1/2 and 1/12 respectively. Often n is taken to be 12 to avoid the square root and division.

Box–Muller Algorithm

This method gives an exact sample. Sample random variables $U_1, U_2 \sim U(0,1)$ and put

$$X_1 = \cos(2\pi U_1)\sqrt{-2\ln U_2},$$
$$X_2 = \sin(2\pi U_1)\sqrt{-2\ln U_2}. \tag{2.18}$$

Then X_1 and X_2 are independent normally distributed random variables with mean 0 and variance 1. The proof of this is given in Figure 2.10.

Although it might seem that knowing X_1 imparts some information about X_2, this is not so; X_2 can still be any real value at all. An example is shown in Figure 2.9. The graph labeled "$X1 = 0.7$" graphs the points (U_1, U_2) in the plane satisfying the first of equations (2.18) for a given value of $X1$, in this case 0.7. The graph labeled "$X2 = 2.0$" shows the points satisfying the second of equations (2.18) for the given value of $X2$. As shown, there is a unique pair (U_1, U_2) giving these choices.

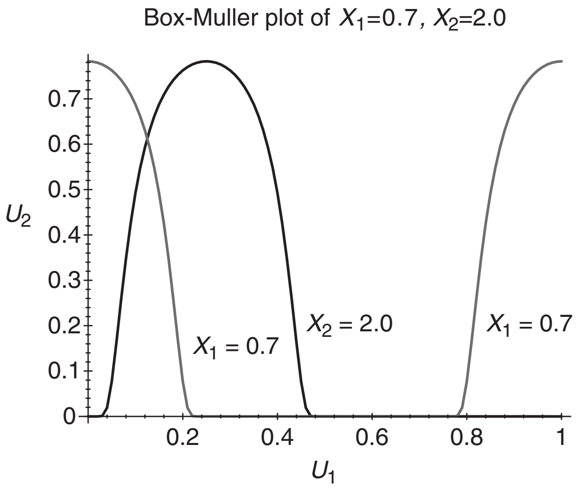


Fig. 2.9. Solving for (U_1, U_2) for a given (X_1, X_2).

Marsaglia–Bray Algorithm

This method also gives two exact samples, similar to the Box–Muller algorithm.

1. Sample random variables $U_1, U_2 \sim U(0, 1)$ and put $V_1 = 2U_1 - 1$, $V_2 = 2U_2 - 1$, and $S = V_1^2 + V_2^2$.

2. if $S \geq 1$ go to 1, else return

$$X_1 = V_1 \left(\frac{-2}{S} \ln S \right)^{1/2},$$

$$X_2 = V_2 \left(\frac{-2}{S} \ln S \right)^{1/2}.$$

Then X_1 and X_2 are independent normally distributed random variables with mean 0 and variance 1.

The Marsaglia–Bray algorithm was derived as an improvement in speed over the Box–Muller algorithm, since it uses only one transcendental function evaluation, while the Box–Muller uses three. This is offset somewhat by the fact that a fraction of the time Step 1 is rejected and must be repeated. (What is this fraction?) Today the speed of the computer and the efficiency of transcendental function evaluation has made these issues moot in most cases.

The proof that the Marsaglia–Bray algorithm gives exact samples from $N(0,1)$ is similar to that of the Box–Muller algorithm and we omit it.

2.6.2 Approximate Sampling via the Central Limit Theorem

Suppose we want a sample of the outcome of 1000 Bernoulli trials with probability of success $p = 0.995$. This might correspond to the number of acceptable light bulbs to come out of a manufacturing process in one hour. Of course we could simulate the 1000 trials. Or we could instead exploit the CLT as follows. Let R_{1000} be the random variable for the number of successes, and let μ and σ^2 refer to the Bernoulli trial random variable; $\mu = p$ and $\sigma^2 = pq$. Then N defined by

$$N = \frac{R_{1000} - 1000\mu}{\sqrt{1000\sigma^2}} = \frac{R_{1000} - 995}{2.2305}$$

is approximately $N(0,1)$. Now let n be a sample from $N(0,1)$, say $n = 0.68$. Then

$$r = 2.2305n + 995 = 996.52 \to 997, \quad \text{rounded to an integer,}$$

is the corresponding sample for R_{1000}.

Proof for the Box–Muller algorithm. Let $f_{X_1,X_2}(x_1,x_2)$ be the joint density of X_1 and X_2 and let $f_{U_1,U_2}(u_1,u_2)$ be the joint density of U_1 and U_2. Then

$$f_{U_1,U_2}(u_1,u_2) = f_{U_1}(u_1)f_{U_2}(u_2) = 1, \quad \text{if } 0 \le u_1, u_2 < 1,$$

and 0 otherwise. Let τ be the change of variable

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \tau(u_1,u_2) = \begin{pmatrix} \cos(2\pi u_1)\sqrt{-2\ln u_2} \\ \sin(2\pi u_1)\sqrt{-2\ln u_2} \end{pmatrix}.$$

Squaring x_1 and x_2 and adding gives

$$x_1^2 + x_2^2 = -2(\ln u_2)\cos^2(2\pi u_1) - 2(\ln u_2)\sin^2(2\pi u_1) = -2\ln u_2.$$

So

$$u_2 = e^{-\frac{1}{2}(x_1^2+x_2^2)}. \qquad (*)$$

This shows that $0 < u_2 \le 1$. The variable u_1 may be similarly restricted due to the periodicity of sine and cosine. Similar to the 1-variable change of variable formulas derived previously, the relationship between the two densities may be written either as

$$f_{X_1,X_2}(\tau(u_1,u_2)) \det \tau' = f_{U_1,U_2}(u_1,u_2)$$

or equivalently as

$$f_{X_1,X_2}(x_1,x_2) = f_{U_1,U_2}(\tau^{-1}(x_1,x_2)) \det(\tau^{-1})'.$$

Since f_{U_1,U_2} is identically 1, and remembering that $\det(\tau^{-1})' = 1/\det\tau'$, the latter gives

$$f_{X_1,X_2}(x_1,x_2) = \det(\tau^{-1})' = 1/\det\tau'.$$

Now

$$\begin{aligned}
\det\tau' &= \det\begin{pmatrix} \frac{\partial x_1}{\partial u_1} & \frac{\partial x_1}{\partial u_2} \\ \frac{\partial x_2}{\partial u_1} & \frac{\partial x_2}{\partial u_2} \end{pmatrix} \\
&= \det\begin{pmatrix} -2\pi\sin(2\pi u_1)\sqrt{-2\ln u_2} & \frac{-(1/u_2)}{\sqrt{-2\ln u_2}}\cos(2\pi u_1) \\ 2\pi\cos(2\pi u_1)\sqrt{-2\ln u_2} & \frac{-(1/u_2)}{\sqrt{-2\ln u_2}}\sin(2\pi u_1) \end{pmatrix} \\
&= \frac{2\pi}{u_2}\sin^2(2\pi u_1) + \frac{2\pi}{u_2}\cos^2(2\pi u_1) = \frac{2\pi}{u_2}.
\end{aligned}$$

From equation $(*)$, $u_2 = e^{-\frac{1}{2}(x_1^2+x_2^2)}$. Therefore

$$\det(\tau^{-1})' = \frac{u_2}{2\pi} = \frac{1}{2\pi}e^{-\frac{1}{2}(x_1^2+x_2^2)}$$

and

$$f_{X_1,X_2}(x_1,x_2) = 1\cdot\frac{1}{2\pi}e^{-\frac{1}{2}(x_1^2+x_2^2)} = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x_1^2}\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x_2^2}.$$

QED.

Fig. 2.10. Proof for the Box–Muller algorithm.

2.6.3 Error Estimates for Monte Carlo Simulations

Suppose we are trying to use Monte Carlo to estimate some value, call it θ. It could be the waiting time of a discrete event simulation or the neutron flux through a thickness of shielding or any scalar value that results from an instance of a simulation.

Let X_1, X_2, …, X_n be n estimates of θ as derived from the outcome of the simulation. If the X_i are independent and identically distributed with mean θ, then by the central limit theorem their sample average $\bar{X}$ is approximately normally distributed with mean equal to θ and variance equal to σ_X^2/n, where σ_X^2 is the (unknown) variance of the X_i. In this case

$$Y = \frac{\bar{X} - \theta}{\sqrt{\sigma_X^2/n}}$$

is approximately $N(0,1)$ distributed. From a $N(0,1)$ table we notice that among other things, with probability 0.954 a normal sample lies within two standard deviations of the mean; hence

$$\Pr\left(-2 < \frac{\bar{X} - \theta}{\sqrt{\sigma_X^2/n}} < 2 \right) = 0.954.$$

In other words, with probability 0.954, θ lies in the interval

$$\bar{X} - 2\sqrt{\sigma_X^2/n} < \theta < \bar{X} + 2\sqrt{\sigma_X^2/n}.$$

Now, given a value for σ_X^2, we may calculate probabilistic error bounds for θ, or *confidence intervals* as they are called.

In practice there are three problems with this program. First, usually σ_X^2 must itself be estimated from the data. In that case Y above will not be normal. Second, the X_i may not be identically distributed; the simulation may suffer start-up effects, for example. And third, the X_i may be correlated. Both of these issues are, in fact, a common difficulty in queuing simulations.

An attempt to deal with the second and third issues is by *batching*. Divide the n trials into m batches each of size J:

$$X_1 \;\; \ldots \;\; X_J \;\mid\; X_{J+1} \;\; \ldots \;\; X_{2J} \;\mid\; \ldots \;\mid\; X_{(m-1)J+1} \;\; \ldots \;\; X_{mJ}.$$

Thus there are $m = n/J$ batches. The batch random variables

$$B_i = \frac{1}{J} \sum_{j=(i-1)J+1}^{iJ} X_j, \qquad i = 1, \ldots, m,$$

tend to be independent and identically distributed. Now we may apply the development above to the B_i in place of the X_i. Thus

$$\hat{\theta} = \bar{B} = \frac{1}{m} \sum_{i=1}^{m} B_i$$

is an estimator $\hat{\theta}$ for θ. And the random variable

$$Y = \frac{\hat{\theta} - \theta}{\sqrt{\sigma_B^2/m}}$$

is approximately $N(0,1)$.

If we knew the variance σ_B^2 of the batch random variables, then we could use the normal distribution itself to make error bounds as was done above. However, σ_B^2 is generally not known and must itself be estimated from the B_i data. The *sample variance* for σ_B^2 is given by

$$s_B^2 = \frac{1}{m-1} \sum_{i=1}^{m} (B_i - \hat{\theta})^2 \tag{2.19}$$

and is itself a random variable (known to be gamma distributed; see Section 2.8).

Thus in place of Y we have

$$t = \frac{\hat{\theta} - \theta}{(s_B/\sqrt{m})}, \tag{2.20}$$

the quotient of a normal random variable by a gamma random variable; see [HT01]. Such a combination is a *Student-t* random variable. The Student-t is well known and has one parameter, its *degrees-of-freedom* or DOF. Tables giving ordinates of the Student-t are included in the appendix of most statistics books. We provide an abbreviated one as well, Table 2.4. So we must use the *t-statistic* to obtain the confidence intervals we seek.

For example, given α, t_α is defined by

$$\Pr(-t_\alpha < t < t_\alpha) = \alpha.$$

These values t_α can be looked up in or derived from a t-table. If the table gives cumulative values instead of two-sided values, as does the table below, a conversion is required. Since the t distribution is symmetric, $\Pr(-t_\alpha < t < t_\alpha) = \alpha$ is equivalent to

$$\Pr(t < t_\alpha) = \alpha + \frac{1-\alpha}{2} = \frac{1+\alpha}{2}.$$

Thus if one wants, say, a 95% confidence interval for t, use the cumulative table with $\beta = (1 + 0.95)/2 = 0.975$.

The Student-t distribution table gives t_β vs DOF, where t_β is defined by $\Pr(t < t_\beta) = \beta$.

DOF	1	2	3	4	5	6	8	10	15	20	30	40	60	120	∞
t_β	6.31	2.92	2.35	2.13	2.02	1.94	1.86	1.81	1.75	1.73	1.70	1.68	1.67	1.66	1.65

Table 2.3. Cumulative t values for $\beta = 0.95$.

DOF	1	2	3	4	5	6	8	10	15	20	30	40	60	120	∞
t_β	12.71	4.30	3.18	2.78	2.57	2.45	2.31	2.23	2.13	2.09	2.04	2.02	2.00	1.98	1.96

Table 2.4. Cumulative t values for $\beta = 0.975$.

A problem that arises with batching is deciding how to divide up the n samples between the number of batch replications m versus the number of repetitions per batch J. We want m as large as possible, since a larger m reduces the size of the confidence interval. This argues for a small value of J. But J must be large enough that the batch random variables tend to be independent and uncorrelated.

2.7 Rejection Sampling Example: The Beta Distribution

The Marsaglia–Bray algorithm for the normal distribution introduces a new idea into sampling, namely a test imposed on candidate values in order to proceed. This idea can be expanded upon to form the basis of a powerful and completely general sampling method, called *rejection sampling.*

Consider the problem of sampling a Bernoulli trial with probability $p = 1/3$ using coin flips. Here is one solution: flip the coin twice; if the outcome is HH, return "success," if HT or TH, return "failure," and if TT, reject the trial and go again. This gives the right probabilities because

$$\begin{aligned}\Pr(\text{return is success}) &= \Pr(\text{return is success}|\text{there is a return})\\ &= \frac{\Pr(\text{there is a return and it is success})}{\Pr(\text{there is a return})}\\ &= \frac{1/4}{3/4} = \frac{1}{3}.\end{aligned}$$

By adding the acceptance/rejection discipline, only the relative values of the target probabilities matter. That is, they could have been 1/7, 1/7, 1/7 and "success" will still be returned 1/3 of the time.

Consider a second example. Using three coin tosses, we want to select "1" with probability 2/5, and "2," "3," and "4" each with probability 1/5. Since tossing a coin twice gives probabilities of 1/4 and this is less than 2/5, we divide the target probabilities by $M = 8/5$ (to bring 2/5 down to 1/4), recalling that only their relative values matter. The modified target probabilities are then 1/4, and 1/8, 1/8, 1/8. To find the acceptance discipline, reason as follows. If the result HH occurs, we accept this all the time and return "1" because $\frac{1/4}{1/4} = 1$. If the result HT occurs, we accept this one-half of the time and map its return to "2"; this gives the correct probability because $\frac{1/8}{1/4} = 1/2$. Similarly we accept TH and TT one-half of the time and map these outcomes to "3" and "4" respectively. If a value is rejected, we start over and toss the two coins again.

The probability of accepting a value is this: all the time for HH, and 1/2 of the time for the others; hence

$$\frac{1}{4}\cdot 1 + \frac{1}{4}\cdot\frac{1}{2} + \frac{1}{4}\cdot\frac{1}{2} + \frac{1}{4}\cdot\frac{1}{2} = \frac{5}{8},$$

or 62.5% of the time; so 37.5% of the time we must start again. When a value is returned, the probability that it is a "1" is given by

$$\begin{aligned}\Pr(\text{returned value is "1"}) &= \frac{\Pr(\text{"1" is selected and returned})}{\Pr(\text{some value is returned})} = \frac{\frac{1}{4}\cdot 1}{\frac{5}{8}}\\ &= \frac{2}{5}.\end{aligned}$$

In the same manner, for "2" we have

$$\Pr(\text{returned value is “2”}) = \frac{\Pr(\text{“2” is selected and returned})}{\Pr(\text{some value is returned})} = \frac{\frac{1}{4}\cdot\frac{1}{2}}{\frac{5}{8}}$$
$$= \frac{1}{5},$$

and the other target probabilities are likewise seen to have the proper frequencies.

To actually carry out the method, toss a coin twice; if the outcome is HH, then return “1”; if the outcome is HT or TH or TT, then we must do a side experiment to decide whether to accept the outcome and return its mapped value or reject the outcome and start again. Since the side experiment is a 50-50 decision, it could itself be a coin toss; say an H means accept and a T means reject. For example, if the third toss is H then return “2” in the HT case, “3” in the TH case, and “4” in the TT case. If it is T, then start all over.

Figure 2.11 illustrates the situation in the example. For each possible outcome from the first two coin tosses, we must scale the resulting probability to obtain the correct value.

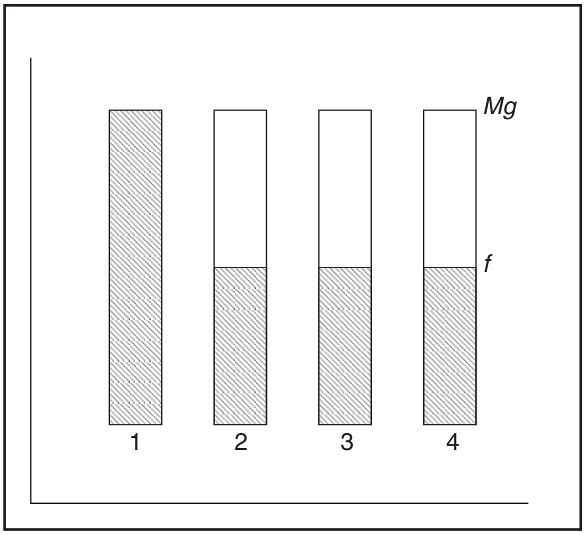


Fig. 2.11. Target f and proposal Mg densities for the example in the Text.

The density actually sampled from, the two coin tosses in the above examples, is called the *proposal density* g. It can be any convenient density that is nonzero where the target density f is nonzero. Let $h(\cdot)$

refer to the acceptance discipline. The *generalized rejection sampling method* consists of two simple steps.

Rejection Sampling

1. Let Y be a sample from pdf g.
2. With probability $h(Y)$ return $X = Y$; otherwise go to (1).

As in the example, step 2 is a Bernoulli trial with success probability $h(Y)$ and is carried out through a side experiment.

Referring again to Figure 2.11, we see that $M = 8/5$ and we must scale the proposal g (in the example, a uniform distribution given by two coin tosses) by 8/5 in order to have $f \le Mg$. Then for each outcome drawn from g, the proportion of f that lies under this part of g gives the acceptance probability. For the first outcome of "1" we see that $f = 2g$, so this one is always accepted. For the rest, $f = (1/2)2g$ (the shaded part represents only 1/2 of the vertical bar at these positions), and so we accept only with probability 1/2.

Rejection sampling is a completely general method for sampling from both discrete and continuous distributions. To see that it works in the continuous case, calculate the infinitesimal probability that the returned value X equals x as follows:

$$\begin{aligned}\Pr(X = x)\,dx &= \frac{\Pr(Y = x \text{ and } x \text{ is returned })}{\Pr(\text{some value is returned})}\\ &= \frac{g(x)\,dx\,h(x)}{\int_{-\infty}^{\infty} g(z)h(z)\,dz}.\end{aligned}$$

Hence the probability density function of returned values is

$$pdf_X(x) = \frac{g(x)h(x)}{\int_{-\infty}^{\infty} g(z)h(z)\ dz}.$$

To construct a sampling algorithm, let f be the target pdf, let g be the proposal pdf, and let the constant M be such that $f(x) \le Mg(x)$ for all x. The acceptance/rejection discipline is

$$h(x) = f(x)/(Mg(x)). \tag{2.21}$$

With this, the pdf of accepted values is

$$\frac{g(x)h(x)}{\int gh} = \frac{f(x)/M}{\int f(z)/M\ dz} = f(x).$$

As above, the probability of an acceptance is

$$\Pr(Y \text{ is accepted}) = \int_{-\infty}^{\infty} g(z)h(z)\ dz = 1/M.$$

This is the efficiency of the generator, so keep M as small as possible. Figure 2.12 illustrates the situation for continuous f and g.

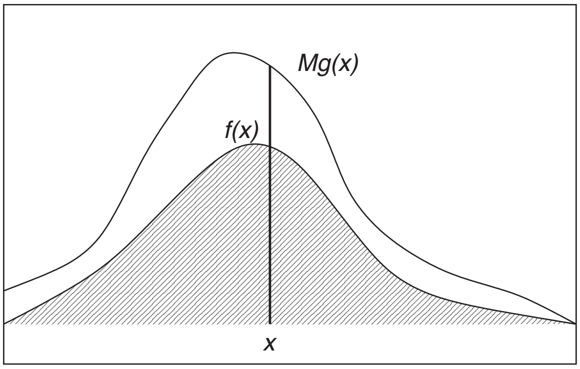


Fig. 2.12. Rejection method in the continuous case.

Notice that it is not a problem if $g(x) = 0$ for some x, since we only use (2.21) to compute $h(x)$ for values of x that are sampled from a distribution with pdf g. The probability of obtaining an x such that $g(x) = 0$ is zero, and thus the situation will never occur.

2.7.1 Sampling from the Beta Distribution

The beta distribution, $\mathrm{Be}(\alpha, \beta)$, is defined as the density given by

$$f(x) = cx^{\alpha-1}(1-x)^{\beta-1}, \quad 0 \le x \le 1, \tag{2.22}$$

where the constant c is chosen to make $\int_0^1 f(x)\,dx = 1$. The parameters α and β can be any positive real values. If either α or β is less than 1, the density is unbounded but the integral is nonetheless finite. The mean of the beta distribution is

$$\mu = \frac{\alpha}{\alpha+\beta}$$

and the variance is

$$\mathrm{var} = \frac{\alpha\beta}{(\alpha+\beta+1)(\alpha+\beta)^2}.$$

Suppose we want to draw samples from Be$(2,3)$ whose density is $f(x) = 12x(1-x)^2$, $0 \le x \le 1$; see Figure 2.13. We will use the uniform $U(0,1)$ as the proposal; thus $g(x) = 1$, $0 \le x \le 1$. Since $f(x) \le 16/9$ (from the figure), we may take $M = 16/9$. Then the acceptance discipline is

$$h(x) = f/Mg = \frac{27}{4}x(1-x)^2. \tag{2.23}$$

The rejection sampling algorithm becomes (1) sample $U \sim U(0,1)$, and (2) with probability $h(U)$ return $X = U$; otherwise go to (1).

A run might go as follows, $U \sim U(0,1) = 0.8213$, say, so $h(U) = 0.0786$ as given by (2.23). Now, sampling $R \sim U(0,1)$ as the side experiment gives, say, $R = 0.616$. This is not less than or equal to 0.0786 so $U = 0.8213$ is rejected. Retrying, this time $U \sim U(0,1) = 0.384$, say, and $h(U) = 0.4371$. Another call to $U(0,1)$ produces $R = 0.00506$, which is less than or equal to 0.4371, so the side experiment succeeds. This time $U = 0.384$ is accepted and is returned.

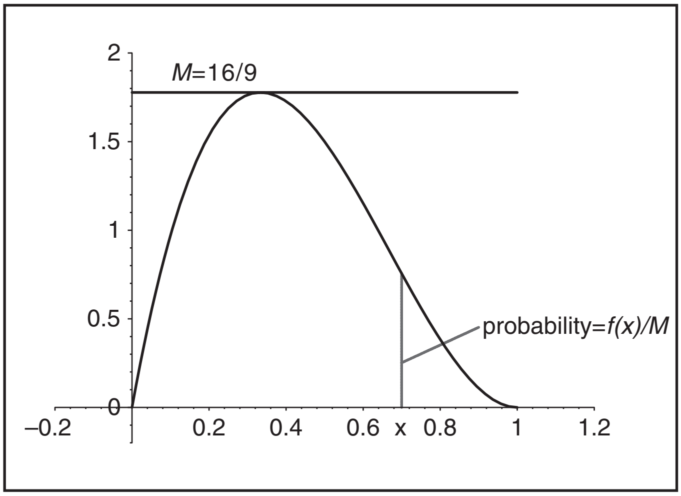


Fig. 2.13. The β-distribution $12x(1-x)^2$.

This example shows that one can view rejection sampling as an example of hit-or-miss. Note that the chosen value U is accepted if $\frac{16}{9}R < f(U)$ and rejected if $\frac{16}{9}R \ge f(U)$. Plot the point $(U, \frac{16}{9}R)$ in Figure 2.13, and we see that U is accepted if it lies below the graph of f and rejected if above (or on) the graph.

2.7.2 Sampling from an Unbounded Beta Distribution

We take up one more example of rejection sampling. This time we seek samples from an unbounded beta distribution. For simplicity take $\alpha = \beta$ with $\alpha < 1$ (so that the density is unbounded at $x = 0, 1$), but the same idea works more generally. The density is

$$f(x) = cx^{\alpha-1}(1-x)^{\alpha-1}, \qquad 0 \le x \le 1,$$

where c is the normalizing factor. Recall that only relative values of f are needed, so it is not necessary to find c. This is an important feature of the rejection sampling method.

As the proposal distribution we introduce a composition dividing the domain into the subintervals $[0, 1/2)$ and $[1/2, 1)$. We will have more to say about compositions in the next section. By symmetry of this $f(x)$ we need only sample X from the interval $[0, 1/2)$ and, with probability 1/2, return either X or $1 - X$. Therefore as the proposal density we use

$$g(x) = c'x^{\alpha-1}, \qquad 0 \le x \le 1/2.$$

The cdf is the integral

$$G(x) = \int_0^x c't^{\alpha-1}\,dt = c'\frac{x^\alpha}{\alpha}.$$

Since $G(1/2) = 1$, we get $c' = \alpha/(1/2)^\alpha$.

To sample from g by cdf inversion entails solving

$$U = \frac{c'}{\alpha}Y^\alpha$$

for Y. We get

$$Y = \left(\frac{\alpha}{c'}U\right)^{1/\alpha} = \frac{1}{2}U^{1/\alpha},$$

where we have substituted for c' from above.

Since $\alpha < 1$, the largest value of $(1-x)^{\alpha-1}$ for $0 \le x \le 1/2$ occurs when $x = 1/2$. Therefore we may take $M = (1/2)^{\alpha-1}c/c'$, and with this choice we have

$$\begin{aligned} Mg(x) &= (1/2)^{\alpha-1}\frac{c}{c'}c'x^{\alpha-1} = (1/2)^{\alpha-1}cx^{\alpha-1} \\ &\ge (1-x)^{\alpha-1}cx^{\alpha-1} = f(x), \qquad 0 \le x \le 1/2. \end{aligned}$$

Finally, the acceptance discipline is given by

$$h(x) = \frac{f(x)}{Mg(x)} = \frac{cx^{\alpha-1}(1-x)^{\alpha-1}}{(1/2)^{\alpha-1}cx^{\alpha-1}} = (2(1-x))^{\alpha-1}\,.$$

As predicted, c drops out and its value is not needed.

The algorithm for $\mathrm{Be}(\alpha, \alpha)$ is then:

1. generate $U \sim U(0,1)$ and put $Y = (1/2)U^{1/\alpha}$;
2. as the side experiment, if $R \sim U(0,1) < h(Y)$ then accept Y and go to step 3; else return to step 1;
3. again sample $U \sim U(0,1)$; if $U < 1/2$ return $X = Y$; otherwise return $X = 1 - Y$.

2.8 Composition Example: The Gamma Distribution

A density f may be constructed from others, say f_1, f_2, …, f_n, by forming the sum

$$f(x) = p_1 f_1(x) + p_2 f_2(x) + \cdots + p_n f_n(x). \tag{2.24}$$

The weights p_1, p_2, …, p_n are positive and sum to 1. The mean of a composition density is

$$\mu = p_1\mu_1 + p_2\mu_2 + \cdots + p_n\mu_n, \tag{2.25}$$

and the variance is given by

$$\mathrm{var} = p_1^2\,\mathrm{var}_1 + p_2^2\,\mathrm{var}_2 + \cdots + p_n^2\,\mathrm{var}_n, \tag{2.26}$$

since we take the constituent distributions f_i to be independent. In this μ_i and var_i are the mean and variance of the ith distribution.

To sample from a composite distribution, pick index i as in roulette wheel selection with probability p_i, for example via discrete cdf inversion or the alias method. Having chosen i, sample from f_i by an appropriate method for that density and return the sampled value.

Composition can give rise to a great variety of probability distributions. One class of examples is the sum of piecewise constant functions such as a histogram. In this case the f_i are just uniform densities and are easily sampled. Any density can be approximated in this way. A better approximation is to use piecewise trapezoidal regions. In this case the f_i are appropriately scaled linear functions and these too are easily sampled.

Another way in which composition can be exploited for sampling purposes is by tailoring a piecewise envelope or proposal density g for use in the rejection method. This gives rise to an exact sampling method. The combination of composition and rejection provides a powerful tool for many distributions that cannot be treated in any other way.

We already saw a restricted example of this approach in the last section, restricted in the sense that by symmetry the two components were essentially the same distribution. Here we show how the composition/rejection technique can provide a solution for treating the gamma distribution. This is another important distribution that arises in conjunction with the Poisson process.

2.8.1 The Gamma Distribution

In the Poisson process with event rate λ we saw that the exponential distribution is the waiting time until the first event. By way of generalization, the *gamma distribution* with parameter α is the waiting time W until the αth event occurs. Using (2.9) we obtain the gamma cdf as follows:

$$\begin{aligned} F(w) &= \Pr(W \le w) = 1 - \Pr(W > w) \\ &= 1 - \Pr(\text{fewer than } \alpha \text{ events occur in } [0,w]) \\ &= 1 - \sum_{k=0}^{\alpha-1} \frac{(\lambda w)^k e^{-\lambda w}}{k!}. \end{aligned} \tag{2.27}$$

Differentiating this with respect to w, we obtain the gamma density function

$$f(w) = \frac{\lambda}{(\alpha-1)!}(\lambda w)^{\alpha-1}e^{-\lambda w}. \tag{2.28}$$

Denote this distribution by $G(\alpha, \lambda)$.

To begin our treatment of the gamma distribution we first simplify it by means of a transformation inspired by our work with the exponential distribution. It is easy to see that if $W \sim G(\alpha, \lambda)$, then $Y = \lambda W$ is distributed as $G(\alpha, 1)$. Therefore we will restrict our attention to $G(\alpha, 1)$ and use the inverse transformation, $W = (1/\lambda)Y$, to get back to the more general version if necessary.

One of the most important properties of the gamma distribution is the *reproductive property*:

if $W_1 \sim G(\alpha_1, \lambda)$ and $W_2 \sim G(\alpha_2, \lambda)$ are independent, then $W_1 + W_2$ is distributed as $G(\alpha_1 + \alpha_2, \lambda)$. (See Figure 2.14 for a proof in the integer case.)

This property is to be expected. The waiting time for α events to occur is just the α-fold sum of the waiting time for the events to occur one by one. By the reproductive property it follows that the mean and variance of the gamma are just the α-fold sums of the mean and variance of the exponential; hence

$$\mu = \frac{\alpha}{\lambda}, \quad \text{var} = \frac{\alpha}{\lambda^2}.$$

As we will see shortly, the reproductive property will also allow us to generate samples from the gamma distribution by sampling exponentials.

The gamma distribution with α an integer is also known as the *Erlang distribution*. The gamma distribution extends the Erlang to the case that α is any positive real number. For nonintegral α the factorial is replaced by the *gamma function*

$$\Gamma(t) = \int_0^\infty y^{t-1} e^{-y}\, dy, \qquad 0 < t. \tag{2.29}$$

For integer arguments $n = 1, 2, \ldots$,

$$\Gamma(n) = (n-1)!.$$

For noninteger arguments the value is given by the integral directly (2.29). In particular, it can be shown that

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}.$$

2.8.2 Sampling from $G(\alpha, 1)$

First let $0 < \alpha < 1$. For purposes of rejection, the target density is

$$f(x) = \frac{1}{\Gamma(\alpha)} x^{\alpha-1} e^{-x}.$$

Since only relative values count, we can ignore the constant $1/\Gamma(\alpha)$. We will create a proposal density g by composition by splitting the domain at $x = 1$. Notice that the value of e^{-x} at $x = 1$ is $1/e$, but the value of $x^{\alpha-1}$ at $x = 1$ is 1, which is e times as big. We need this observation to stitch together the components of g.

If W_1 is distributed as $G(\alpha_1, 1)$ and W_2 as $G(\alpha_2, 1)$ and they are independent, then $W_1 + W_2$ is distributed as $G(\alpha_1 + \alpha_2, 1)$.
Decompose the event $W_1 + W_2 = w$ according to W_1; in infinitesimals we have

$$\begin{aligned}\Pr(W_1 + W_2 = w) &= \sum_{0<x<w} \Pr(W_1 + W_2 = w \,|\, W_1 = x)\Pr(W_1 = x) \\ &= \sum_{0<x<w} \Pr(W_2 = w - x)\Pr(W_1 = x) \\ &= \int_0^w \frac{(w-x)^{\alpha_2 - 1}}{(\alpha_2 - 1)!} e^{-(w-x)} \frac{x^{\alpha_1 - 1}}{(\alpha_1 - 1)!} e^{-x}\, dx \\ &= \frac{e^{-w}}{(\alpha_2 - 1)!(\alpha_1 - 1)!} \int_0^w (w-x)^{\alpha_2 - 1} x^{\alpha_1 - 1}\, dx.\end{aligned}$$

Suppose α_1 is an integer; then use integration by parts, letting $u = x^{\alpha_1 - 1}$ and $dv = (w-x)^{\alpha_2 - 1}\, dx$. The integral becomes

$$\int_0^w \frac{(w-x)^{\alpha_2 - 1 + 1}(\alpha_1 - 1)x^{\alpha_1 - 1 - 1}}{(\alpha_2 - 1 + 1)}\, dx.$$

Continue performing integration by parts, a total of n times, until $\alpha_1 - 1 - n = 0$, i.e., $n = \alpha_1 - 1$ times. Then we will have

$$\begin{aligned}\Pr(W_1 + W_2 = w) &= \frac{e^{-w}}{(\alpha_2 - 1 + n)!} \int_0^w (w-x)^{\alpha_2 - 1 + n}\, dx \\ &= \frac{w^{\alpha_2 + n} e^{-w}}{(\alpha_2 + n)!} = \frac{w^{\alpha_1 + \alpha_2 - 1} e^{-w}}{(\alpha_1 + \alpha_2 - 1)!}.\end{aligned}$$

Fig. 2.14. Derivation of mean and variance of the Poisson distribution.

Let f_1 be the distribution on $[0, 1)$ defined by

$$f_1(x) = \alpha x^{\alpha - 1}, \qquad \mathrm{cdf}_1(x) = x^\alpha.$$

And let f_2 be the distribution on $[1, \infty)$ defined by

$$f_2(x) = e^{1-x}, \qquad \mathrm{cdf}_2(x) = 1 - e^{1-x}.$$

Let g be the composite density

$$g(x) = pf_1(x) + (1-p)f_2(x).$$

It will be convenient to define $\chi_1(x)$ as the *characteristic function* of $[0, 1)$,

$$\chi_1(x) = \begin{cases} 1 & \text{if } 0 \le x < 1, \\ 0 & \text{otherwise,} \end{cases}$$

and $\chi_2(x)$ as the characteristic function of $[1, \infty)$,

$$\chi_2(x) = \begin{cases} 1 & \text{if } 1 \le x, \\ 0 & \text{otherwise.} \end{cases}$$

Then g may be written

$$g(x) = p\alpha x^{\alpha-1}\chi_1(x) + (1-p)e^{1-x}\chi_2(x),$$

where $0 < p < 1$ is our choice.

For reasons that will be clear in a moment, choose p such that $\lim_{x\uparrow 1} g(x) = e \lim_{x\downarrow 1} g(x)$. This gives

$$p\alpha = e(1-p), \qquad \text{so } \; p = \tfrac{e}{e+\alpha} \; \text{ and } \; 1 - p = \tfrac{\alpha}{e+\alpha}.$$

With this, g becomes

$$g(x) = \frac{e\alpha}{e+\alpha}\left(x^{\alpha-1}\chi_1(x) + e^{-x}\chi_2(x)\right).$$

Now we must find M such that $Mg \ge f$.

For $0 \le x \le 1$ the maximum value of e^{-x} is 1; hence choose M such that

$$M\frac{e\alpha}{e+\alpha}x^{\alpha-1} = x^{\alpha-1} \ge x^{\alpha-1}e^{-x}.$$

This gives

$$M = \frac{e+\alpha}{e\alpha}.$$

This value of M works for $x \ge 1$ as well; thus

$$Mg = M\frac{e\alpha}{e+\alpha}e^{-x} = e^{-x} \ge x^{\alpha-1}e^{-x},$$

where we note that the maximum value of $x^{\alpha-1}$ is 1 for $x \ge 1$.

The acceptance discipline is

$$\frac{f(Y)}{Mg(Y)} = \begin{cases} e^{-Y} & \text{on } [0,1), \\ Y^{\alpha-1} & \text{on } [1,\infty). \end{cases}$$

Altogether, the algorithm is:

1. let $U_1, U_2 \sim U(0,1)$,
2. if $U_1 < \frac{e}{e+\alpha}$, case A, put $Y = U_2^{1/\alpha}$; otherwise, case B, put $Y = 1 - \ln U_2$;

3. as the side experiment, let $R \sim U(0,1)$; in case A, return $X = Y$ if $R < e^{-Y}$; in case B, return $X = Y$ if $R < Y^{\alpha-1}$; otherwise go to step 1.

The efficiency of the algorithm is

$$1/M = \frac{e\alpha}{e+\alpha}.$$

Now let $\alpha > 0$ be arbitrary and let $m = \text{int}[\alpha]$ and $\delta = \text{frac}(\alpha)$. By the reproductive property, if $Y \sim G(m,1)$ and $Z \sim G(\delta,1)$, then $X = Y + Z \sim G(\alpha,1)$. So it remains to show how to generate samples from the Erlang distribution $G(m,1)$. But this is easily accomplished by simulation. In Section 2.5.1 we saw that $W = -\ln U$, where $U \sim U(0,1)$ is a sample from $E(1)$; therefore $-\sum_1^m \ln(U_i)$, $U_i \sim U(0,1)$, is a sample from $G(m,1)$. This may be simplified slightly:

1. for $i = 1, 2, \ldots, m$ let $U_i \sim U(0,1)$;
2. return $Y = -\ln(\prod_{i=1}^m U_i)$.

The returned value Y is distributed like $G(m,1)$.

Notes

The problem of devising algorithms for sampling from various distributions has attracted considerable interest. The book [Rub81] has a wealth of algorithms for many distributions. Probably the biggest effort in this regard is the two-volume treatise published by the Los Alamos National Laboratory based on work done at the lab shortly after the Monte Carlo method was invented.

There is also a large interest in obtaining random samples of various types of combinatorial objects such as graphs, permutations, partitions, and tilings. The Markov chain Monte Carlo method (see Chapter 3) has both been greatly influenced by and been very useful in this search.

Multivariate distributions are very common, and thus it is a common problem to try to extract structure from data derived from a multivariate distribution. This is the central problem in the area of *data mining*, and many different ways have been devised to graphically investigate high-dimensional data sets. Clearly there is no one correct way of doing this.

Problems: Chapter 2

1. (4) (St. Petersburg problem) (a) Player A pays house B \$5 and flips a coin. If it falls tails, the game ends. If it falls heads, B gives A \$1 and A flips the coin again. This continues until the coin lands tails, with B's

payoff to A doubling for each successive head. Simulate this game 100 times, then 1000 times, then 10,000 times and histogram A's winnings. (b) Keep track of A's fortune during the play of the series of games. What observations can you make about this situation?

2. (5) (Paris Salon problem) (a) Let X be the random variable for the first time a 6 is obtained on the roll of a fair die. This should take 6 rolls on average, should it not? Simulate this experiment (many times) to obtain a histogram and to estimate the expectation of X. (b) Given the results of (a) would you bet \$1 to make \$1 that a 6 will not be obtained in 4 rolls of a fair die? Simulate this experiment and report.

3. (3) Let g be the piecewise constant density

$$g(x) = \begin{cases} c, & 0 \le x < 0.1, \\ 2c, & 0.1 \le x < 0.7, \\ c/3, & 0.7 \le x < 1.0, \end{cases}$$

and 0 elsewhere. Find c such that this is a probability density function. Show how to sample from g by inverting the cdf.

4. (4) (Composite density) Let f be the piecewise linear function given by

$$f(x) = \begin{cases} x/4, & 0 \le x < 1, \\ 1/4, & 1 \le x < 3, \\ (x-3)/60 - (x-6)/12, & 3 \le x < 6, \\ 1/20 & 6 \le x < 15, \end{cases}$$

and 0 elsewhere. Find the constant c such that $g(x) = cf(x)$ is a probability density. What are the respective probabilities that a sample falls within the regions A $[0, 1)$, B $[1, 3)$, C $[3, 6)$, and D $[6, 15)$? Devise a method for sampling from g and implement it.

5. (4) Let g be the function on $[-2, 2]$ given by

$$g(x) = \frac{8}{7} + \frac{118}{63}x^2 - \frac{74}{63}x^4 + \frac{10}{63}x^6.$$

Find the constant c such that $f(x) = cg(x)$ is the pdf for a distribution. Show how to sample from this density.

6. (5) (Points in a box) N points are placed uniformly at random in the unit square. What is the expected minimum distance between them? Simulate the problem for different values of N and graph the distance versus N.

7. (4) (Sampling from a multivariate normal) If $X_1, X_2 \sim N(0, 1)$, argue that the vector $\mathbf{X} = (X_1, X_2)$ has a distribution whose pdf has circular symmetry. In fact, the pdf is the function (where C is a normalizing constant)

$$f(x_1, x_2) = Ce^{-(x_1^2+x_2^2)/2}.$$

This is an example of a *bivariate normal distribution*. Now, given a 2×2 matrix A and a vector $\boldsymbol{\mu} = (\mu_1, \mu_2)$, let

$$(Y_1, Y_2) = \mathbf{Y} = A\mathbf{X} + \boldsymbol{\mu}$$

and investigate the distribution of $\mathbf{Y}$.

8. (4) (Random point from a sphere) The objective of this problem is to devise a way to find a random sample from the sphere

$$S = \{(x_1, x_2, \ldots, x_n) : x_1^2 + x_2^2 + \cdots + x_n^2 = 1\}$$

in $\mathbb{R}^n$. The hint to do this is to find a multivariate distribution in $\mathbb{R}^n$ whose pdf f has spherical symmetry (that is, $f(\mathbf{x}) = f(\mathbf{y})$ if $\mathbf{x}$ and $\mathbf{y}$ are the same distance from the origin; see Problem 7). Then sample a point $(X_1, X_2, \ldots, X_n)$ from f, and the required sample should be

$$\frac{(X_1, X_2, X_3, \ldots, X_n)}{(X_1^2 + X_2^2 + \cdots + X_n^2)^{1/2}}.$$

Try out your solution in $\mathbb{R}^2$ (obtaining points on the unit circle) and histogram them along the line segment $[0, 2\pi]$.

9. (4) (Distribution of the minimum) Fix $N = 10$, for the moment. Choose independently $X_i \sim U(0,1)$ for $i = 1, 2, \ldots, 10$; the goal is to find the distribution of $\min X_i$. That is, we want to know $\Pr(\min X_i \le t)$ for each $t \in [0, 1)$. Do this by simulation, that is, generate multiple trials of this experiment and histogram the random variable $\min X_i$. How do the results depend on N? To see this, change N and repeat.

10. (4) (Distribution of the minimum) Assuming that the X_i are independent, argue that

$$\Pr(\min X_i > t) = \prod_i \Pr(X_i > t)$$

to compute the pdf and cdf of the distribution of $\min X_i$, where $X_i \sim U(0,1)$.

11. (4) Show that if X and Y are independently distributed with cdfs F_X and F_Y respectively, then $Z = \max\{X, Y\}$ has cdf

$$F_Z(z) = F_X(z)F_Y(z).$$

Use this to find an algorithm to sample from

$$Z = \max\{U_1, U_2, \ldots, U_n\},$$

where each $U_j \sim U(0,1)$.

12. (3) Obtain samples from a Poisson random variable with parameter $\lambda = 0.1$ per minute for $t = 30$ minutes by sampling from (a) $b(20, \frac{\lambda t}{20})$, (b) $b(60, \frac{\lambda t}{60})$, and (c) $b(100, \frac{\lambda t}{100})$. Histogram the results of all three and compare. Keep track of run times.

13. (3) Obtain samples from a Poisson random variable with parameter $\lambda = 0.1$ by "Method 2" from Section 2.5.1 and histogram the results. Compare with the results from the previous problem.

14. (4) (Sampling bias for bus waiting times) Suppose the interarrival time for a city bus has an exponential distribution with parameter $1/\lambda$. A passenger arrives at a uniformly random time and records the time until the next bus arrives. What is the expected waiting time? Use a simulation to get an answer. Is the answer surprising? Now suppose instead that the interarrival time is $U(0, 2\lambda)$. How does this change the situation? (Notice that the expected interarrival time is λ in both cases.)

15. (2) Devise an algorithm to obtain samples from the distribution on $[0, 1]$ with pdf $f(x) = nx^{n-1}$ for any fixed natural number n.

16. (5) (Difference between equally likely and conditional equally likely; see [Ross96] page 67) This problem examines a situation in which exponential waiting times yield an "equally likely" distribution. Consider an exponential distribution with parameter λ and a situation in which exactly $n = 50$ events occurred in the interval $[0, 1]$. Compare this situation to one obtained by choosing 50 samples U_i from $U(0, 1)$ as the events.
For one side of the comparison, let U_i be 50 samples from $U(0, 1)$. For example, suppose the random numbers are

$$0.48,\quad 0.36,\quad 0.66,\quad 0.12,\quad 0.82,\quad 0.57,\quad 0.28,\quad 0.11,\quad 0.18,\quad 0.71.$$

Now sort them to get

$$0.11,\quad 0.12,\quad 0.18,\quad 0.28,\quad 0.36,\quad 0.48,\quad 0.57,\quad 0.66,\quad 0.71,\quad 0.82,$$

and histogram the interpoint distances

$$\begin{aligned}
&(0.12 - 0.11 =)0.01, \quad (0.18 - 0.12 =)0.06, \quad (0.28 - 0.18 =)0.10,\\
&(0.36 - 0.28 =)0.08, \quad (0.48 - 0.36 =)0.08, \quad (0.57 - 0.66 =)0.11,\\
&(0.66 - 0.71 =)0.04, \quad (0.82 - 0.71 =)0.11.
\end{aligned}$$

To get an estimate of the distribution, this needs to be done many times. For the other side of the comparison it is necessary to generate $E(\lambda)$ waiting times that yield exactly 50 events in the interval $[0, 1]$. The parameter λ should be set so that the expected number of events is $n = 50$. The simplest (but certainly not the most efficient) method to do this is by rejection, that is, generate samples from $E(\lambda)$ and check whether exactly 50 fit in the interval $[0,1]$. How inefficient is this? That is, how many rejections does one get per acceptance? Can you come up with another method?

17. (3) Show that the following "zombie gap" algorithm approximately simulates exponential arrivals. Find the exponential parameter. (From *Scientific American* Computer Recreations, April 1985.) Let the "gap" be from 0 to w, $w < 1$.

```
loop:
  count $= 0$
  loop:
    increment count
    choose $U \sim U(0,1)$
    if( $U < w$ ) break out of loop
    endloop
  report count as next arrival time
endloop
```

Here is a sample run: $2, 4, 1, 1, 3, 1, 2, 5, \ldots$, meaning that the first event occurred at time 2, the next at time 6 (interarrival time of 4), etc. Show that the interarrival times are approximately exponentially distributed, especially for small w. What is the exponential parameter?

18. (3) Suppose a species of bacteria reproduce with exponentially distributed waiting times having parameter $\lambda = 2$. Further suppose there is a "survival reward" associated with a birth at time t given by $r(t) = te^{-t}$. By simulation, estimate the expectation of r. What is the exact value? (Hint, $\int_0^\infty te^{-t}\lambda e^{-\lambda t}dt = (\lambda/(\lambda+1)^2) = 2/9$.)

19. (6) (Pure birth process.) Suppose a newly born bacterium waits an exponentially distributed period of time T with rate parameter λ, in short $T \sim E(\lambda)$, and then divides, replacing itself with two newly born bacteria. Simulate the growth of such a colony starting from a single bacterium for 60 minutes assuming $\lambda = 0.05$ per minute. What is the (empirical) doubling time for such a colony? What is the average number of generations in this period of time (60 minutes)? (Note: one can derive the differential equations for a birth process exactly as in the Poisson derivation, the only difference being that at the time there are k cells, then the event rate parameter is $k\lambda$ instead of just λ; hence the system of differential equations becomes $dP_k(t)/dt = -\lambda k P_k(t) + \lambda(k-1)P_{k-1}(t)$, and their solution is $P_k(t) = e^{-\lambda t}(1-e^{-\lambda t})^{k-1}$, $k = 1, 2, \ldots$, where $P_k(t) = \Pr(X(t) = k)$. The expectation is $\mathbb{E}(X(t)) = e^{\lambda t}$, i.e., exponential growth. Hence the doubling time for a colony of bacteria is the solution to $\ln(2) = \lambda t$, while for a single bacterium, it is the solution to $1 = \lambda t$ via the Poisson expectation

$\mathbb{E}(X(t)) = \lambda t$.)

20. (5) If $T_\lambda \sim E(\lambda)$ and, independently, $T_\mu \sim E(\mu)$, what is the probability that $T_\lambda < T_\mu$? (Hint, the joint pdf is $\lambda\mu e^{-\lambda r} e^{-\mu s}$.)

21. (7) (Birth and death process.) When a cell is "born," draw a sample $T_b \sim E(\lambda)$ and a sample $T_d \sim E(\mu)$. If $T_b \le T_d$, then the simulated cell divides at time T_b into two new cells as above (and T_d is discarded). If $T_d < T_b$, then the cell dies (and T_b is discarded). Simulate this process for 20 minutes starting from a single cell with $\mu = 1$ per minute and (a) $\lambda = 1$, (b) $\lambda = 1.05$, and (c) $\lambda = 1.10$ all in units of per minute. Compute the empirical mean and variance of the number of cells at the end of the study.

22. (7) (Single-server queue.) People join a queue according to the exponential distribution $E(\lambda)$. The service time for the person at the front of the queue is exponential $E(\mu)$. Simulate the process for 30 minutes for several runs. Experiment with different values of λ and μ; note that if μ is not close to λ, then the queue grows exponentially. Graph the length as a function of time for one of the runs. Find the mean and variance of the waiting time in the queue.

23. (7) (Lanchester battlefield model.) Let B denote the number of blue soldiers and R the number of red soldiers. Blue soldiers have killing effectiveness parameter μ_B and red soldiers have effectiveness μ_R. The Blue army starts with B_0 soldiers and the Red army with R_0. The waiting time for the next death of a Blue soldier is exponential $E(R\mu_R)$, where R is the number of Red soldiers at that time and the waiting time for the death of a Red soldier is similarly exponential $E(B\mu_B)$. To start, sample $T_B \sim E(R_0\mu_R)$ and $T_R \sim E(B_0\mu_B)$. Move time forward to each event, handle it, and schedule the next. Thus, if $T_B < T_R$, move time forward to T_B, decrement B by one, and resample $T_B \sim E(R\mu_R)$. And continue until one army or the other is wiped out. Fix $\mu_R = 1$ and $R_0 = 100$ and experiment with values of μ_B and B_0. In each case find the (mean) time of the battle and each side's expectation of winning. Plot the battle strength vs time for both armies for a typical run.

24. (5) (Retirement benefit projection) At age 50 Fannie Mae has \$150,000 invested and will be investing another \$10,000 per year until age 70. Each year the investment grows according to an interest rate that is normally distributed with mean 8% and standard deviation 9%. At age 70, Fannie Mae then retires and withdraws \$65,000 per year until death. Below is given a conditional death probability table. Thus if Fannie Mae lives until age 70, then the probability of dying before age 71 is 0.04979. Simulate this process 1000 times and histogram the amount of money Fannie Mae has at death.

Mortality table, probability of dying during the year by age[*]							
50	0.00832	64	0.02904	78	0.09306	92	0.26593
51	0.00911	65	0.03175	79	0.10119	93	0.28930
52	0.00996	66	0.03474	80	0.10998	94	0.31666
53	0.01089	67	0.03804	81	0.11935	95	0.35124
54	0.01190	68	0.04168	82	0.12917	96	0.40056
55	0.01300	69	0.04561	83	0.13938	97	0.48842
56	0.01421	70	0.04979	84	0.15001	98	0.66815
57	0.01554	71	0.05415	85	0.16114	99	0.72000
58	0.01700	72	0.05865	86	0.17282	100	0.76000
59	0.01859	73	0.06326	87	0.18513	101	0.80000
60	0.02034	74	0.06812	88	0.19825	102	0.85000
61	0.02224	75	0.07337	89	0.21246	103	0.90000
62	0.02431	76	0.07918	90	0.22814	104	0.96000
63	0.02657	77	0.08570	91	0.24577	105	1.0000

[*] Source: Society of Actuaries, Life Contingencies.

25. (4) (Median of exponential vs mean) Suppose the lifetime of lightbulbs is exponential with mean 1000 hours. Given a randomly selected lightbulb, what is the probability that it will still be burning after 1000 hours? Perform this experiment 100 times and histogram the event "less than 693 hours" versus "more than 693 hours." (The *median* m is the point where $\Pr(T \le m) = \Pr(T > m)$, so $m = \text{cdf}^{-1}(1/2)$. For the exponential, $m = \ln(2)/\lambda = \mu \ln 2$.)

26. (4) Implement all three of the normal sampling routines (via the central limit theorem, Box–Muller algorithm, Marsaglia–Bray algorithm). Test them for (a) speed, (b) accuracy (make a pdf and cdf for all three). Compare the results and give some conclusions.

27. (3) One way to check an RNG is to make a *correlation plot*. This is a plot of sequential pairs (X_n, X_{n+1}) generated by the RNG. Ideally, there should be no structure to this plot. Make a correlation plot for a sequence of normal pseudorandom variables using either the Box–Muller or Marsaglia–Bray algorithm.

28. (4) Test the central limit theorem by sampling (a) 10 times, (b) 100 times from an exponential distribution; form the normalized random variable; and plot a pdf and cdf. Compare with table values for the cdf, see Table 2.2. Repeat with several different values of the parameter λ. How does λ influence the outcome?

29. (6) Repeat the previous problem except use several exponential, gamma, uniform, Poisson, and beta distributions. How does the choice of distribution affect the outcome? How does the choice of parameter for the

distribution effect the outcome? Does it make much difference whether the distribution is symmetric or not?

30. (3) Show that if $X \sim N(0,1)$, then $Y = \sigma X + \mu$ is distributed as normal with mean μ and variance σ^2.

31. (5) (Lognormal distribution.) Let X be normally distributed with mean α and variance β and let $X = \ln Y$ (that is, $Y = e^X$). (a) Find the probability density function of Y. (b) Devise a way to sample from this distribution, i.e., to produce random variables Y with the correct distribution.

32. (4) Let g be the density $g(x) = (x-2)/12 - (x-6)/24$, $2 \le x < 6$, and 0 elsewhere. Sample from g by (a) inverting the cdf and (b) rejection. Histogram and compare the speed of drawing N samples for a convenient N (100,000 or 1,000,000 perhaps).

33. (3) Adapt the rejection method to sample from a *beta* distribution whose pdf is $f(x) = cx^{\alpha-1}(1-x)^{\beta-1}$, $\alpha > 0$, $\beta > 0$, and $0 \le x \le 1$. The constant c is chosen so that $\int_0^1 f(x)dx = 1$. Use $\alpha = 5$ and $\beta = 2$, for example. Notice that the value of c is not needed (except to compute the optimal value for M). Make a histogram.

34. (5) Devise an envelope function g to use in the rejection method to sample from the distribution whose pdf is $f(x) = cx^2e^{-x}$, $0 \le x < \infty$, where c is chosen so that $\int_0^\infty f(x)dx = 1$. Test your samples for accuracy with the pdf by histogramming, check for correlation (by graphing (x_n, x_{n+1})), and test for efficiency (i.e., rate of generation) and the effect of the starting point (by doing a pdf starting from different points).

35. (6) Do a pure birth model, see Exercise 19 above, using a gamma distribution in place of the exponential distribution for waiting times.

36. (7) Do a birth and death model, see Exercise 21 above, using a gamma distribution in place of the exponential distribution for waiting times.

37. (4) Can you think of an alternate way of obtaining a probability $p = 1/3$ Bernoulli trial random variable (or any other for $0 < p < 1$) using coin flips? (Hint: (thanks to Michael Lacy) express p in binary.)

3

Markov Chain Monte Carlo

3.1 Discrete Markov Chains

A *Markov chain* extends the idea of a single probabilistic experiment on the outcome space Ω to a sequence of experiments on Ω, one for every $t = 0, 1, \ldots$ Letting X_t denote the tth outcome, we say that the process moves from state X_{t-1} to state X_t on the tth iteration. The other major novelty here is that the probabilities governing the next move can depend on the present state. In fact, it is usually the case that from any given state x it is possible to move to only a small subset of Ω, called the *neighborhood* of x.

It turns out that this setup of a recurring probabilistic process has wide applicability. Some examples are the changes from moment to moment of a thermodynamic system, the changes in a species' DNA sequence wrought by mutations, the step-by-step folding of a protein molecule, the day-to-day price of a stock, a gambler's fortune from gamble to gamble, and many others. The crucial aspect of a Markov chain is that the system must evolve from one moment to the next in a random way, but depending only on the state of the system at the given moment and not on the entire history.

As the process is performed repeatedly, what conclusions can be drawn about it?

3.1.1 Random Walk on a Graph

A Markov chain can be illustrated by a kind of graph as follows. The states are depicted as the *vertices* of the graph. A *directed edge* extends from one vertex, x say, to another, y, if a transition from x to y is possible in one iteration. See Figure 3.1. The set of all states that can be reached from x in one iteration is the *neighborhood* of x, denoted by N_x. Each such directed edge has a probability p_{xy} associated with it. This is the probability that the edge will be chosen on the next iteration when the chain currently is in the state x.

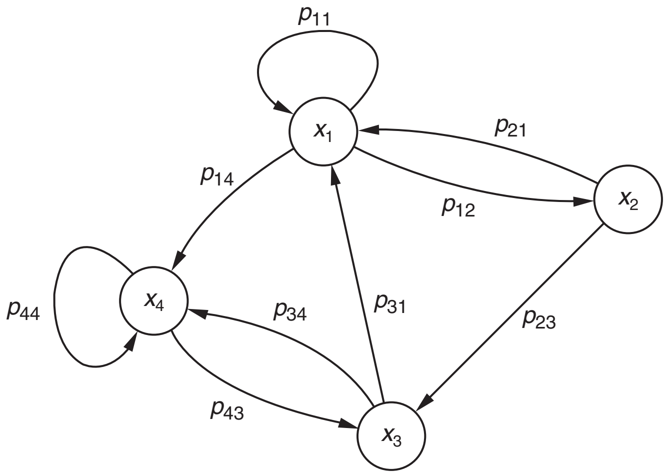


Fig. 3.1. The graph representation of a four-state Markov chain. Its corresponding transition matrix is

$$\begin{bmatrix} p_{11} & p_{12} & 0 & p_{14} \\ p_{21} & 0 & p_{23} & 0 \\ p_{31} & 0 & 0 & p_{34} \\ 0 & 0 & p_{43} & p_{44} \end{bmatrix}.$$

The chain starts in some state, say X_0, which may be chosen randomly according to a *starting distribution* or just assigned. From then on, the chain moves from state to state on each iteration according to the neighborhood transition probabilities. Given that the chain is in state x on some iteration, the next state is chosen according to the discrete density given by $\{p_{xy} : y \in N_x\}$. Notice that the sum of the outgoing probabilities from each vertex x must be 1 to cover all the possible transitions from x,

$$\sum_{y \in N_x} p_{xy} = 1, \qquad \text{for all } x \in \Omega. \tag{3.1}$$

The selection is implemented using any of the roulette wheel selection methods. In the graph, the chain moves along the chosen edge to the next chosen vertex (or stays put if the edge joins the vertex to itself), and the process is ready for the next iteration. In this way, a Markov chain can be envisioned as a random walk on a graph (we will have much more to say about random walks in Chapter 5).

To illustrate, consider the following hypothetical model for the stock market: relative to yesterday, today the market can be up, down, or unchanged as follows (see Figure 3.2):

- if yesterday was up, then the probabilities for today are up 0.3, unchanged 0.2, down 0.5;
- if yesterday was unchanged, then the probabilities for today are up 0.4, unchanged 0.2, down 0.4;
- if yesterday was down, then the probabilities for today are up 0.4, unchanged 0.3, down 0.3.

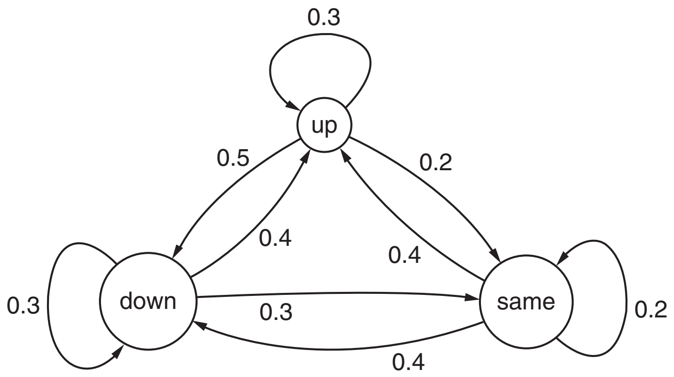


Fig. 3.2. The stock market Markov chain example.

Simulating a Small Chain

To simulate the stock market chain, we first decide on a starting state. We could make the selection randomly or just assume some state deterministically. For our example let us say that yesterday was "up" and we want to simulate today's market. To do so we must make a roulette wheel style selection among "up," "unchanged," and "down" with probabilities 0.3, 0.2, and 0.5 respectively. For this we may use the methods of Section 2.1, inverting the cdf, or Section 2.2, the alias method. For cdf inversion the cumulative break points are 0.3, 0.5, and 1.0; make a side experiment for $U \sim U(0, 1)$. If $U < 0.3$ we stay in "up," if $0.3 \le U < 0.5$ move to state "unchanged," and if $U \ge 0.5$ move to state "down."

To carry this out in software, it is helpful to construct a cumulative distribution matrix C for the chain by rows:

$$C = \begin{array}{r} \\ \text{up} \\ \text{same} \\ \text{down} \end{array} \begin{array}{c} \begin{array}{ccc} \text{up} & \text{same} & \text{down} \end{array} \\ \begin{bmatrix} 0.3 & 0.5 & 1.0 \\ 0.4 & 0.6 & 1.0 \\ 0.4 & 0.7 & 1.0 \end{bmatrix} \end{array}. \tag{3.2}$$

Then for the simulation loop, take the following:

```
select $i$ // get the chain started in state $i$
Loop over repetitions $t = 1, 2, \ldots$
  $U \sim U(0,1)$ // a random value in $[0,1)$
  $j = 1$
  while( $c_{ij} \le U$)
    $j = j + 1$
  end while
  // $j$ becomes the next state, histogram it
  $i = j$ // update for the next iteration
end Loop
```

As the algorithm runs it reports an *instance* of the chain, namely the sequence of states X_t, $t = 1, 2, \ldots$, that occurred as a result of the particular random numbers that were chosen.

Alternatively, we could histogram a run. In this way we get the fraction of time the chain spends in each state. If the histogram is replotted after every repetition t, we will see that it changes considerably at first but eventually settles down and becomes nearly constant. This demonstrates one of the fundamental properties of a regular Markov chain, the existence of an *invariant distribution.* If two simple regularity conditions are satisfied (see the next section), a Markov chain will, asymptotically, visit all its states, at random to be sure, but nevertheless each with a specific frequency.

Thus in our market model, if we knew these frequencies, we would know whether "up" occurred more often than "down" and be able to predict the drift of the market.

For the most part we are not interested in particular instances of a chain; rather we are interested in its general properties, properties valid over all instances. Since the chain is a random process, these properties will naturally be probabilistic.

For this we need to histogram in a different way. Suppose we start our stock market chain in the "up" state and perform exactly one repetition, that is, sample the random variable X_1. Do this many times and histogram the result; it is a histogram of the distribution of X_1. If we could do this infinitely many times, the histogram would show "up" at 0.3, "unchanged" at 0.2, and "down" at 0.5 because these are the transition probabilities out of the "up" state, and thus will be the corresponding proportions.

Similarly, we can do the same thing for X_t for any $t > 0$. We are led to an important tool in understanding Markov chains, the *state probability vector*, or in brief, the probability vector p_t. It is the probability distribution of X_t. Understanding the behavior of p_t is a major goal in understanding a Markov chain; in particular, the long-term behavior of p_t is of interest. By convention we take p_t to be a row vector.

If we let X_0 represent the random variable for the starting state, then p_0 is the starting distribution. For example, it could be $p_0 = [\,1 \quad 0 \quad 0\,]$, meaning a deterministic start in the "up" state.

3.1.2 Matrix Representation of a Chain

As an alternative to the graph representation, a Markov chain can be defined by the matrix $P = (p_{xy})$ of the probabilities p_{xy} of transitioning from state x to state y in one repetition of the chain. This is called the *probability transition matrix* or just *transition matrix* in brief. By (3.1), each row of P must sum to 1.

But this matrix formulation is much more than a tabular description of the chain; it is also a calculation tool. Let p_t be the probability vector of the random variable X_t. Then we can calculate p_{t+1} as the matrix product

$$p_{t+1} = p_t P, \qquad t = 1, 2, \ldots. \tag{3.3}$$

For example, the transition matrix of our stock market example is

$$P = \begin{array}{c} \\ \text{up} \\ \text{same} \\ \text{down} \end{array} \begin{array}{c} \begin{array}{ccc} \text{up} & \text{same} & \text{down} \end{array} \\ \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \end{bmatrix} \end{array}.$$

Starting the chain in the "up" state, we may calculate p_1,

$$p_1 = [\,1 \quad 0 \quad 0\,] \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \end{bmatrix} = [\,0.3 \quad 0.2 \quad 0.5\,],$$

which is the same as we got for the distribution of X_1 above. Continuing,

$$p_2 = p_1 P = \begin{bmatrix} 0.3 & 0.2 & 0.5 \end{bmatrix} \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.32 & 0.32 & 0.36 \end{bmatrix},$$

and so on for any value of t recursively. It is clear by recursion that

$$p_t = p_0 P^t. \tag{3.4}$$

Continuing with the example above, the next few distributions p_t are

$$\begin{aligned} p_3 &= \begin{bmatrix} 0.332 & 0.304 & 0.364 \end{bmatrix}, \\ p_4 &= \begin{bmatrix} 0.3304 & 0.303 & 0.3664 \end{bmatrix}, \\ p_5 &= \begin{bmatrix} 0.33032 & 0.3036 & 0.36608 \end{bmatrix}, \\ \dots & \qquad \dots \\ p_{10} &= \begin{bmatrix} 0.330357 & 0.303571 & 0.366072 \end{bmatrix}. \end{aligned}$$

Notice how the distribution quickly settles down and tends to a limit, the invariant distribution. This is a general feature of Markov chains.

Invariant Distribution

For regular Markov chains, the probability vectors p_t tend toward the invariant distribution π in the limit. This distribution earns its name because

$$\pi P = \pi. \tag{3.5}$$

In other words, if at any time the probability vector becomes the invariant distribution, then it remains so from then on out.

The proof of the existence of and convergence to the invariant distribution is outlined in Appendix B, and is given by the Perron–Frobenius theorem. The conditions on P we have been calling regular are these:

1. *irreducible*: this means, in the graph representation, that there is a directed path leading from every vertex to every other vertex.
2. *aperiodic*: this means that there is no integer $d > 1$ such that the number of iterations between the return from any state to itself must be a multiple of d; see Figure 3.3 for an example of a chain that is periodic.

The first condition is a type of mixing condition on the chain, implying that the only (nontrivial) invariant region is the entire state space. Clearly, if this condition is not satisfied (as when the corresponding graph is disconnected into several components), there can be no unique invariant distribution that is always the limit of the probability vector p_t.

If the second condition is violated (as in Figure 3.3), then p_t will oscillate and thus won't converge.

In the small stock market example above, it is not hard to solve the equation $\pi P = \pi$ to find that (exactly)

$$\pi = \left[\begin{matrix} \frac{37}{112} & \frac{34}{112} & \frac{41}{112} \end{matrix}\right] \approx [\, 0.33035714 \quad 0.30357143 \quad 0.36607143 \,],$$

as we saw above.

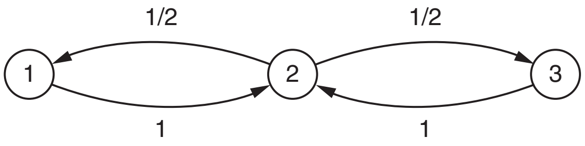

$$\begin{tikzcd} 1 \arrow[r, bend right, "1"'] & 2 \arrow[l, bend right, "1/2"'] \arrow[r, bend left, "1/2"] & 3 \arrow[l, bend left, "1"] \end{tikzcd}$$

Fig. 3.3. A periodic chain of period $d = 2$. For any state x, the number of iterations between returns to x is a multiple of 2.

3.2 Markov Chain Monte Carlo Sampling—The Metropolis Algorithm

The idea for Markov chain Monte Carlo (MCMC) sampling goes back to 1953, when N. Metropolis et al. published the paper "Equations of state calculations by fast computing machines" [MRRTT]. The authors were trying to solve problems in physics that arise due to the random kinetic motions of atoms and molecules. The measurable quantities of the system are expectations over the distribution of the locations, orientations, and velocities of the atoms or molecules. The paper introduced to the world the incredibly useful *Metropolis simulation algorithm*. In fact, this algorithm has been cited as among the top ten algorithms having the greatest influence on the development of science and engineering.

The Metropolis algorithm is at the heart of most MCMC sampling methods. The basic idea behind MCMC is to construct a Markov chain whose invariant distribution is the desired sampling distribution. While this might seem quite complicated, the flexibility inherent in choosing the transition probabilities makes this simpler than it appears.

Our description and motivation of the Metropolis algorithm start with the rejection sampling method. Recall that in the rejection sampling method we have a target density f, a proposal density g, and an acceptance discipline h. Assume, for convenience, that f is a discrete probability density with outcome space Ω. Samples from f will be approximated by the sequence of states $X_1, X_2, \dots$ of a Markov chain over Ω. As in the rejection method, generating the next state is a two-part process: a proposal followed by acceptance or rejection. By regarding the process as a chain, transitions from state to state may depend on the value of the current state. Thus, both the *proposal process* and the *acceptance process* can depend on the current state.

This means that $g_x(\cdot)$ must be defined for every $x \in \Omega$, but in fact, this is often easy to do, and is guided by the particular application (or distribution) in question. In the simplest case, $g_x = g$ of the rejection method for all x, but then we get nothing new (and the associated Markov chain will be very simple, with each iteration independent of the previous one).

The ability to use a different proposal for each point $x \in \Omega$ gives a new range of possibilities over rejection sampling and is the source of the power of the Metropolis algorithm. The one restriction imposed on g_x by the Metropolis algorithm is that it should be symmetric, that is, that

$$g_x(y) = g_y(x). \tag{3.6}$$

While this condition seems restrictive, in most cases it is natural. It is possible to get by with the slightly less restrictive condition that

$$g_x(y) > 0 \text{ if and only if } g_y(x) > 0.$$

This requires some simple modifications of the acceptance discipline that is presented here.

The increased flexibility of using a different proposal for each x comes at a cost, however, since the samples which are generated may now be correlated.

The other difference compared to the rejection method is that the acceptance discipline also depends (in general) on the current state x

as well as the proposed state y. In fact, the acceptance discipline for the Metropolis algorithm is

$$h(x, y) = \min\left(1, \frac{f(y)}{f(x)}\right). \tag{3.7}$$

This says that if the proposed state y has greater probability than the current state x, $f(y) \geq f(x)$, then accept it with certainty. If not, then accept it with probability equal to the ratio $f(y)/f(x)$. As before with the rejection method, only the relative values of f are needed.

Here is the Metropolis algorithm:

```
select $x$ // get the chain started in state $x$
Loop over repetitions $t = 1, 2, \ldots$
  select $y$ from $N_x$ using density $g_x$ // propose $y$
  put $h = \min(1, f(y)/f(x))$ // acceptance probability
  if $U \sim U(0,1) < h$ then $x \leftarrow y$ // else $x$ unchanged
  // histogram $x$
end Loop
```

Recall that the proposal process g has to satisfy the symmetry requirements given in (3.6).

While this may seem simple, it is incredibly powerful and has been generalized and used in a multitude of applications.

3.2.1 Some Examples

To start with something familiar, our first example is simulating dice rolls. Here the state space is $\Omega = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$. To contrast the effect of allowing proposals to be state-dependent, we solve the problem twice, once using the smallest possible neighborhood system and then again using the largest.

Minimal Neighborhoods

Here the neighborhood for each state is just one or two other states. For each $3 \leq x \leq 11$ let

$$g_x(y) = \begin{cases} 1/2, & \text{if } y = x - 1, \\ 1/2, & \text{if } y = x + 1, \\ 0, & \text{otherwise,} \end{cases}$$

and let $g_2(3) = g_2(2) = 1/2$ and $g_{12}(11) = g_{12}(12) = 1/2$. This completes the definition of the proposal functions. Evidently it too may be codified by a matrix, the *proposal matrix*

$$G = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & & & \cdots & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 \end{bmatrix}. \tag{3.8}$$

The first row says that if the current state is $X = 2$, then the proposed next state is 2 or 3, with both equally likely. Thus the first row represents $g_2(\cdot)$. Similarly for the other rows. Notice that we have made the proposal matrix symmetric.

The return values will obviously be correlated; for example, if the present state is 5, the next state will be either 4 or 6, since only these are even proposed. The value of MCMC sampling lies in its ability to generate the correct statistics, provided that a sufficiently large number of repetitions are performed.

Figure 3.4 represents 20,000 repetitions of the MCMC sampling algorithm using this proposal scheme. The figure was generated by the Matlab code given below.

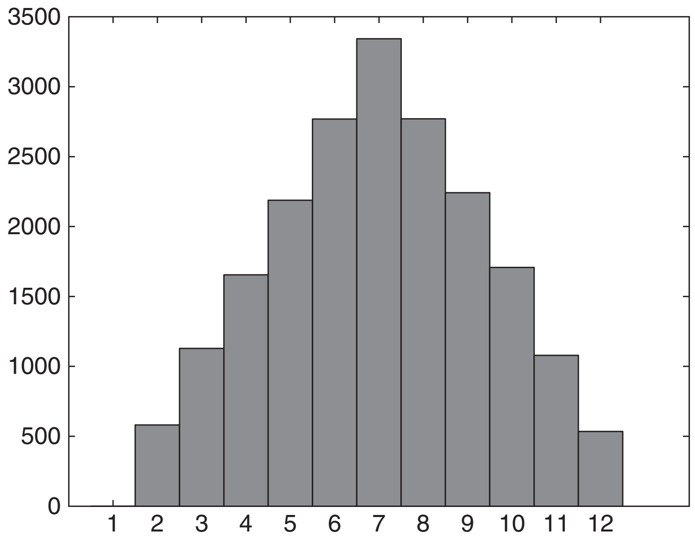


Fig. 3.4. MCMC dice rolls, 20,000 samples.

```
Matlab
> f=[0, 1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1];
> d=zeros(1,20000); % for histogramming
> x=5;
> for i = 1:20000
    U = rand;
    if x == 2
      if U < 0.5
        y = 3;
      else
        y = 2;
      end
    elseif x == 12
      if U < 0.5
        y = 11;
      else
        y = 12;
      end
```

```
continued...
    else
      if U < 0.5
        y = x-1;
      else
        y = x+1;
      end
    end
    h = min(1,f(y)/f(x));
    U = rand;
    if U < h
      x = y;
    end
    d(i) = x; % record the state
  end
> a = 1:1:12;
> hist(d,a)
> [N,h] = hist(d,a)
```

Maximal Neighborhoods

The maximal neighborhood is the one for which $N_x = \Omega$ for all x. That is, from any state x, any other state may be proposed, for example, all with equally likelihood. Such a proposal matrix for dice-rolling is given by

$$G = \begin{bmatrix} 1/11 & 1/11 & \cdots & 1/11 \\ 1/11 & 1/11 & \cdots & 1/11 \\ \vdots & \vdots & \ddots & \vdots \\ 1/11 & 1/11 & \cdots & 1/11 \end{bmatrix}.$$

With this neighborhood choice the proposal distribution, $g_x(\cdot)$, is constant with respect to x. However, the acceptance discipline as given by (3.7) is still bivariate. Notice that this choice of neighborhood corresponds to generating independent samples.

Markov Chain Monte Carlo for Continuous Chains

In this next example we show how the method can be applied to sample from a continuous distribution, namely the gamma distribution. The underlying Markov chain is continuous with state space $\Omega = [0, \infty)$.

To implement the sampling algorithm, note that we have an end-effect problem at the origin. Namely, points $y < 0$ cannot be accepted, but at the same time, the proposal density must be symmetric, $g_x(y) = g_y(x)$. We handle this by adopting as the proposal density, for all x, the uniform density over the interval of length 2 centered at x and simply not accepting any proposed point less than 0. Since $g_x()$ is a uniform density of length 2 centered at x, if y belongs to this interval, then x

belongs to the interval of length 2 centered at y and is proposed just as likely as y is from $g_x()$. Any asymmetry is pushed off to the acceptance discipline. The situation is similar to that in the discrete case above, where the end effect was solved by a self-loop at $x = 2$ and $x = 12$ of probability 1/2.

With that background, the algorithm is simply

```
select $x > 0$ // get the chain started in state $x$
Loop over repetitions $t = 1, 2, \ldots$
  select $y$ from $U(x-1, x+1)$ // propose $y$
  if $y < 0$ then $y \leftarrow x$ // stay at $x$
  $h = \min(1, f(y)/f(x))$ // acceptance probability
  if $U \sim U(0,1) < h$ then $x \leftarrow y$ // else $x$ unchanged
  // histogram $x$
end Loop
```

The resulting frequency diagram is shown in Figure 3.5.

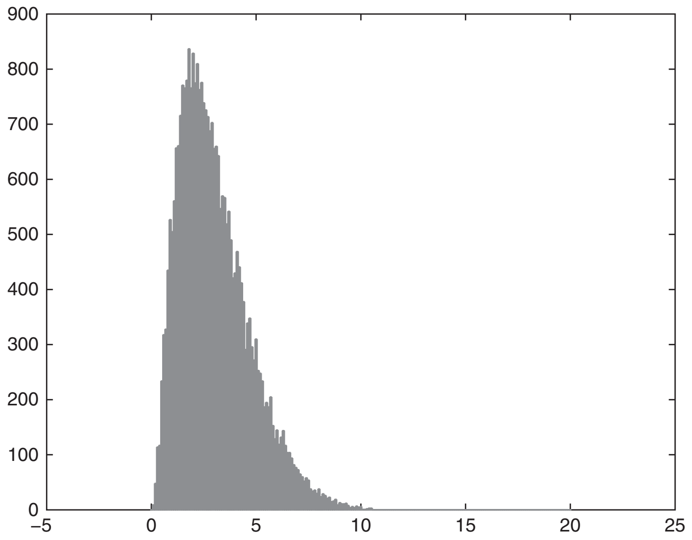


Fig. 3.5. MCMC sampling for $f = 0.5x^2e^{-x}$, 20,000 samples.

3.2.2 Why Does the Metropolis Algorithm Work?

Why does the Metropolis algorithm constitute a Markov chain whose invariant distribution is f?

We have already seen that a regular Markov chain has a unique invariant distribution π and that the probability vector p_t converges to it asymptotically. A good analogy for this is to imagine that there is 1 unit of "probability mass," something like sand, and on each repetition of the chain some of the probability mass of a state flows out along the directed edges of its graph representation and some flows in. The amount that flows out is calculated as the amount of probability mass $p_t(x)$ in x at time t times the fraction that goes to y from x, summed over all y:

$$\text{“mass” out of state } x \text{ on repetition } t = \sum_{y \in \Omega} p_t(x) p_{xy}.$$

And the amount that flows in from other states is

$$\text{“mass” into state } x \text{ on repetition } t = \sum_{y \in \Omega} p_t(y) p_{yx}.$$

For the invariant distribution these are equal for all states x:

$$\begin{aligned}\pi(1)p_{1x} &+ \pi(2)p_{2x} + \cdots + \pi(N)p_{Nx} \\ &= \pi(x)p_{x1} + \pi(x)p_{x2} + \cdots + \pi(x)p_{xN} \\ &= \pi(x)\left(p_{x1} + p_{x2} + \cdots + p_{xN}\right) \\ &= \pi(x). \end{aligned} \tag{3.9}$$

Of course, this is no more than restating $\pi P = \pi$ for the xth row.

But suppose each pair of states exchanges probability mass in balance with each other, this condition is called *detailed balance*,

$$\pi(x)p_{xy} = \pi(y)p_{yx}, \qquad x, y \in \Omega. \tag{3.10}$$

Then (3.9) will follow. In fact, for the Metropolis algorithm, the transition probability is given by

$$p_{xy} = g_x(y)h(x, y),$$

because to make the transition from x to y, y must be proposed and then accepted. Assume that the target distribution f is the invariant distribution and check whether detailed balance holds,

$$f(x)g_x(y)h(x, y) = f(y)g_y(x)h(y, x).$$

Assuming without loss of generality that $f(y) > f(x)$, this becomes

$$f(x)g_x(y)1 = f(y)g_y(x)\frac{f(x)}{f(y)}.$$

Since we assumed that g is symmetric, these are the same, and thus detailed balance (3.10) holds for the Metropolis chain with respect to f. Since there is always at least one state at which it is possible to stay (any y where $f(y) \geq f(x)$ for all x), the chain cannot be periodic. To ensure that the chain is irreducible, it is only necessary to assume that the proposal process governed by g is irreducible.

Under these conditions, the Metropolis algorithm asymptotically tends to the target density f as the number of repetitions increases.

3.3 MCMC Sampling and the Ergodic Theorem

Just knowing that the distribution of a chain converges to the invariant distribution is not quite enough to justify the use of MCMC sampling. This fact alone says only that we can draw a sample from the invariant distribution by waiting long enough and then sampling the position of the chain. However, MCMC sampling computes expectation values (in general), and so requires many samples from the invariant distribution. By the construction of a Markov chain, the samples are not independent, but are correlated. Is this sufficient to be able to compute expectation values? Clearly arbitrary correlated samples are not sufficient (they could all be identical), but those generated by a Markov chain are not arbitrary.

Under the same conditions (irreducibility and aperiodicity) given previously, a Markov chain will have the ergodic property given in the next theorem.

Theorem 3.1. *Let (X_n) be an irreducible aperiodic Markov chain with finite state space Ω and stationary distribution π. Let $r : \Omega \to \mathbb{R}$ be any function. Then*

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} r(X_i) = \mathbb{E}_\pi(r),$$

where by $\mathbb{E}_\pi$ we mean the expectation with respect to the distribution π.

This theorem justifies the use of MCMC in computing expectation values, which is the standard use of MCMC.

3.4 Statistical Mechanics

In this section we give a basic introduction to the principles of statistical mechanics. Our purpose is to provide the foundation for the Ising model, which we take up in the next section, and to explain the origin of the Boltzmann factor in the Metropolis algorithm (whose presence is not evident in the algorithm as we presented it in the previous section). In fact, the original derivation of the Metropolis algorithm stems from the need to produce samples from the Boltzmann distribution for the purpose of solving statistical mechanics problems. *Readers with limited time may safely skip this section with no loss of continuity.*

Statistical mechanics, or statistical physics, is the study of the macroscopic properties of systems having an astronomical number of states of existence. The prototypical example is a gas in a confined region of space. An accurate model must keep track of each gas molecule in all its minute detail: its position, momentum (velocity), quantum state, spin, and so on. Even an approximate model, called *particles-in-a-box*, noting only position and momentum, requires six coordinates per molecule. A cubic meter of air might contain 10^{25} molecules, and thus the state of this system is described by a vector x with 6×10^{25} coordinates.

Such a *microstate* x is a complete specification of the system. Every property of the system can be determined from it. The *phase space* Ω of a system is the space of all possible microstates x. For the particles-in-a-box it encompasses all possible positions and velocities of the particles and is thus astronomically large indeed, albeit finite owing to quantum mechanics.

One of the simplest possible systems is a lattice of N points arrayed in space with each point existing in one of two configurations. Such a system is called the *Ising model*. The configuration of each point is termed its *spin*, and it is either "up" or "down." Mathematically, spin is taken to be $+1$ or -1. The Ising model actually approximates certain real physical systems, magnetic materials for example. Every electron can be considered as a small magnet with either a spin up or a spin down. Electrons within an atom that are up/down paired cancel each other's magnetism. However, a material composed of atoms having unpaired electrons can exhibit magnetism at ordinary scales. The points of the Ising model play the role of atoms with unpaired electrons.

Macroscopic Observables

In aggregate, the entities of the system determine physical properties that can be measured at ordinary scales, the *macroscopic observables* of the system. For the particles-in-a-box, pressure and temperature are examples. In the Ising model, an observable is the magnetic moment of the system defined as the net number of up spins over down spins.

Due to its thermal energy, the microstate of any system is constantly changing in a random manner. As a result, the sequence of microstates executes a random walk through phase space. But the range of macroscopic properties represented by phase space is vast and quite pronounced. For example, it is possible for the particles-in-a-box to be located entirely in one corner of the box occupying a volume 1/10 that of the whole. In the Ising model all of the spins could be up.

These specific states may be just as likely or even more likely than any other specific states, so why don't we ever find all the air in a room packing itself into one corner, or iron spontaneously magnetizing?

Macroscopic Partitions of Phase Space

To answer this question, imagine the states of the phase space partitioned into regions corresponding to the macroscopic property of interest. With regard to particles-in-a-box, one partition region could be for microstates having molecules distributed throughout the box, that is, in every 1/10 subvolume of the box, and another for those microstates putting molecules only in one specific corner, Ω_{corner}, of the box (and many other partition sets as well).

Now we must count the number of microstates in each partition set. If the box has only one particle, and if we ignore velocity, then the size of Ω_{corner} is one-tenth the size of Ω. If there are two particles, then $|\Omega_{\text{corner}}|$ is approximately 1/100th that of $|\Omega|$ (approximate because both particles cannot occupy the exact same location). And if there are 10^{25} particles, then the relative size of Ω_{corner} is approximately

$$p = \left(\frac{1}{10}\right)^{10^{25}} = 10^{-10^{25}}.$$

This is also the chance that all 10^{25} particles will at some time be in the corner of the box. The expected number of samples of Ω in order to get a "hit" in Ω_{corner} is $1/p$, or

$$10^{10^{25}}.$$

If states are sampled at the rate of one billion per second, that is 10^9 every second, then in the entire age of the universe, thought to be about 10^{18} seconds, only 10^{27} points of the phase space have been sampled at most. This is insignificant by comparison.

The situation is similar in the Ising model. The number of states having all spins in the same direction is 2, all up or all down, while the size of the phase space is 2^N. For a 100×100 lattice of spins, a toy system in comparison to any magnetic material, the size of phase space is $2^{10,000} = (2^{10})^{1000} \approx (10^3)^{1000}$ (exactly 1.995×10^{3010} to four decimal places). This is far larger than 10^{80}, the estimated number of protons in the universe.

By contrast, the number of microstates having a net spin of 0, thus a magnetic moment of 0, is

$$\binom{N}{N/2} = \frac{N!}{(N/2)!(N/2)!}$$

when N is even. For a 100×100 lattice, this amounts to some 1.59179×10^{3008} (compare this to the total number of states given above).

As an aside, we work out an approximation to combinatorial numbers such as this; an approximation we will also need later on. It can be derived from the central limit theorem approximation to the binomial random variable B equal to the number of up-spins in the lattice. Take $p = q = 1/2$; then $\mu = np = n/2$ and $\text{var} = npq = n/4$, and

$$\begin{aligned}\binom{n}{x}\frac{1}{2^n} &= \binom{n}{x}\frac{1}{2}^x\frac{1}{2}^{n-x} = \Pr(B = x)\\ &= \Pr\left(x - \frac{1}{2} < B < x + \frac{1}{2}\right).\end{aligned}$$

But by the central limit theorem, approximately $B \sim N(\frac{n}{2}, \frac{n}{4})$. Combining these, we get

$$\begin{aligned}\binom{n}{x}\frac{1}{2^n} &\approx \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{n/4}}e^{-\frac{1}{2}\frac{(t-n/2)^2}{n/4}}\,dt\\ &\approx \sqrt{\frac{2}{\pi n}}e^{-\frac{2}{n}(x-n/2)^2} \cdot 1,\end{aligned}$$

in which it is assumed that the integrand is approximately constant over the interval of length 1. Hence

$$\binom{n}{x} \approx 2^n\sqrt{\frac{2}{\pi n}}e^{-\frac{2}{n}(x-n/2)^2}, \tag{3.11}$$

and so

$$\binom{N}{N/2} \approx 2^N \sqrt{\frac{2}{\pi N}}.$$

For $N = 100$ the approximation is 1.59183×10^{3008}. The relative chance of observing all spins up or down to observing an equal number of up and down spins is approximately

$$\frac{2}{2^N \sqrt{\frac{2}{\pi N}}},$$

a very small number even for modest N.

Entropy

The considerations above show that one of the key properties of thermal systems is the size of the partitions of phase space. Since these numbers are so big, the logarithm, to base e, of size is used in place of the raw count. *Entropy* is defined as

$$S = k_B \log n, \tag{3.12}$$

where k_B is a constant that depends on the units being used and n is the number of states in the partition being counted. Summarizing the above,

> thermal systems tend to be in the macrostates having maximal entropy.

An energy E is always associated with the microstates of a thermal system. For particles-in-a-box, the sum of the kinetic energies $\frac{1}{2}mv^2$ of its particles is a component of the energy of the system. There may be others. In the Ising model, the sum of the interaction energy $-Jx_ix_j$ between every pair of neighboring spins i and j is one component of the system's energy. In this expression, the constant J measures the strength of the interaction. And recall that x_k is either $+1$ or -1 for all components of x. If the system is in an external magnetic field of strength H, then the sum of each spin's interaction with the field, $-Hx_i$, is another other component of the energy.

Systems in Contact

Now consider two systems brought into contact in such a way that they can exchange energy only (not particles). This situation vastly increases the number of microstates available to the combined system. If N_i denotes the number of microstates of system $i = 1, 2$, then $N = N_1 N_2$ is the number of microstates N of the combined system. In terms of entropy this is

$$S = k_B \log(N_1 N_2) = k_B(\log N_1 + \log N_2) = S_1 + S_2. \tag{3.13}$$

So entropy is conveniently additive. Energy is additive as well: the combined energy is the sum of the energies of the individual systems,

$$E = E_1 + E_2. \tag{3.14}$$

What changes can be predicted for the combined system? The answer is that energy will be exchanged in such a way that the entropy of the combined system is maximized. By (3.14), $E_2 = E - E_1$, showing that as energy flows into system 1, the same amount flows out of system 2, so the entropy of the combined system is a function of E_1 only (or E_2 only if we prefer). Maximizing this with respect to E_1 gives

$$0 = \frac{dS}{dE_1} = \frac{dS_1}{dE_1} + \frac{dS_2}{dE_2}\frac{dE_2}{dE_1} = \frac{dS_1}{dE_1} - \frac{dS_2}{dE_2}.$$

Therefore we see that in the final state, the quantity that is equilibrated between the two is dS_i/dE_i. Define the *temperature* T of a system according to the equation

$$\frac{dS}{dE} = \frac{1}{T}. \tag{3.15}$$

Thus in the final state, $T_1 = T_2$. Temperature is taken as the reciprocal of the derivative, so the direction of energy flow will be from the system with high temperature to the system with low temperature.

It is clear that the entropy of the final state of the combined system will be greater than or equal to that of the sum of entropies of the separate systems. This is because the separate systems were constrained to have the energies E_1 and E_2 respectively (for a total of E), but the combined system is constrained only by E and can have any values of E_1 and E_2 (so long as their sum is E).

Boltzmann Factor

Suppose a system under study, system 1, is in contact with a *heat bath*, a much larger system, denoted by 2, tending to keep the smaller one at a constant temperature. We are interested in the probability that the smaller system will be in a given macrostate M as a function of the energy of that macrostate. We know that this is proportional to the number of microstates $N_1 + N_2$ of the combined system for that macrostate. But the heat bath is so much larger that changes to N_2 dominate, while those of the smaller system are negligible in comparison. By expanding the dependence of $S_2(E_2)$ on E_2 in a Taylor series, and using (3.14), we have

$$\begin{aligned} S_2(E - E_1) &= S_2(E) - \frac{dS_2}{dE_2}\, E_1 + \text{higher-order terms in } E_1 \\ &\approx S_2(E) - \frac{1}{T}\, E_1. \end{aligned}$$

Since $S_2 = k_B \log N_2$, or equivalently, $N_2 = e^{\frac{1}{k_B} S_2}$, we have

$$\Pr(M) \propto N_2 = e^{\frac{1}{k_B} S_2} = e^{\frac{1}{k_B} S_2(E)} e^{-\frac{E_1}{k_B T}} \tag{3.16}$$

(the notation $a \propto x$ means that $a = cx$ for some c). The term $e^{-E_1/(k_B T)}$ is the *Boltzmann factor*. It shows that low-energy configurations are more likely than high-energy ones and exactly how this is related to temperature.

The Partition Function

The probability distribution of microstates over Ω can now be given. Its density is

$$f(x) = \frac{e^{-E(x)/k_B T}}{Z}, \tag{3.17}$$

where Z is the normalizing factor. It is called the *Boltzmann distribution.* The normalizing factor is of course

$$Z = \sum_{x \in \Omega} e^{-E/k_B T}. \tag{3.18}$$

But Z, called the *partition function*, is more than just a normalizing factor. It plays a central role in the derivation of the thermodynamic properties of the system. A discussion of these matters leads beyond the scope of this text. Our interest lies with simulating thermal systems and estimating the values of macro observables. For that we need the following definition and the Metropolis algorithm.

Thermal Averages

The value of a macro observable can be predicted by forming its expectation over Ω using the Boltzmann density. Denoting an expectation by the notation $\langle \cdot \rangle$ (a standard notation in statistical mechanics), the expectation of macro observable M is given by

$$\langle M \rangle = \sum_{x \in \Omega} M(x) e^{\frac{-E(x)}{k_B T}} / Z. \tag{3.19}$$

3.5 Ising Model and the Metropolis Algorithm

In this section we carry out an Ising model simulation. Its central feature is the use of the Metropolis algorithm for ensuring that the configurations are sampled from the Boltzmann distribution, at least asymptotically.

Often the purpose of such a simulation is to calculate some macroscopic observable. In this case we will be calculating the magnetic moment M of the system at various temperatures. This is the average of the net up-spin of a configuration averaged over phase space Ω under the Boltzmann distribution, equation (3.17),

$$\langle M \rangle = \frac{1}{Z} \sum_{x \in \Omega} M(x) e^{-\frac{E(x)}{k_B\ T}}. \tag{3.20}$$

To describe the phase space, imagine a lattice Λ with a *spin* at each node. In the Ising model, the spin can take one of two values, ± 1, at each node. A choice of a spin for each node in the lattice is a configuration or microstate $x \in \Omega$. For $x \in \Omega$, $M(x)$ is defined to be the sum of the spins in this particular configuration. Thus, if there are about as many positive as negative spins in x, we would have $M(x)$ close to 0. In (3.20), the number k_B is *Boltzmann's constant*, T is the temperature, and Z is the normalization factor for the Boltzmann distribution. For notational simplicity, often we use

$$\beta = 1/(k_B T). \tag{3.21}$$

In general, the energy function E is given by an expression of the form

$$E(x) = -\frac{1}{2} \sum_{i \in \Lambda} \sum_{j \in \text{neighbors of } i} x_i x_j. \tag{3.22}$$

Thus, having many nodes with spins similar to its neighbors makes the energy decrease.

We will implement a 2-dimensional model, so on a 2D regular grid, with *periodic boundary condition*, meaning that spins in the last column of the domain are neighbors of those in the first column and those in the bottom row of the domain are neighbors of those in the first row. Topologically this forces the lattice onto the surface of a torus (or a doughnut).

An alternative to the periodic boundary condition is the *free boundary condition*, in which the spins along the left boundary of the lattice have only three neighbors, above, below and on the right; those on the right boundary have only those above, below, and on their left, and so on. The corner spins have only two neighbors.

Given a lattice point (i,j), not a boundary point, its neighbors are the four points: $(i-1,j)$, $(i,j-1)$, $(i,j+1)$, and $(i+1,j)$.

The acceptance discipline for the algorithm specializes to the form originally given by Metropolis. From (3.17) for the Boltzmann density,

$$\frac{f(y)}{f(x)} = \frac{e^{-E_y/k_BT}/Z}{e^{-E_x/k_BT}/Z} = e^{-(E_y-E_x)/k_BT},$$

and from (3.7),

$$h = \min(1, e^{-(E_y-E_x)/k_BT}). \tag{3.23}$$

Note that the normalizing factor Z has dropped out.

The spins are initialized in some fashion to begin the run. This distribution is unlikely to be a sample from the Boltzmann distribution. The Metropolis algorithm will asymptotically approximate the Boltzmann distribution as the run proceeds. Therefore statistics should not be kept from the first block of repetitions, but only from some point onward. It is not known how long this block is; the length must be experimentally determined.

Algorithm:

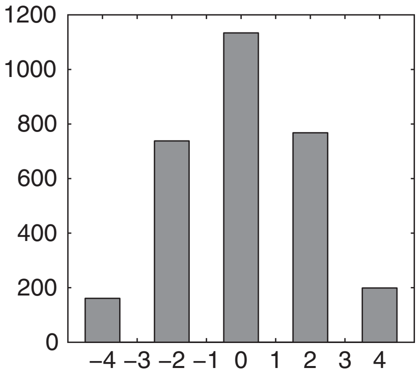
```
initialize $\beta$; // a run at fixed temperature
initialize $x$;  // $L \times L$ lattice of spins
calculate $E_x$; // Energy for $x$
$M = 0$;         // initialize magnetic moment to 0
loop over repetitions $t = 1, 2, \ldots$ startup+nTrials
  select $(i,j) \in L \times L$ equally likely; // a spin to flip
  $y \leftarrow x$; //copy $x$ into $y$
  $y_{ij} = -x_{ij}$; // perturbation of $x$; $y$ is proposed
  calculate $E_y$; //Energy for $y$
  $h = \min(1, e^{-\beta(E_y-E_x)})$ // acceptance probability
```

```
    if $U \sim U(0,1) < h$ then $x \leftarrow y$ // $x$ becomes $y$
      // else $x$ unchanged
    if $t >$ startup then
      calculate $M_x$  // magnetic moment of $x$
      $M \leftarrow M + M_x$ // update the running sum
    endif
  endloop
  $\langle M \rangle$ per spin = $(M/\text{nTrials})/L^2$
```

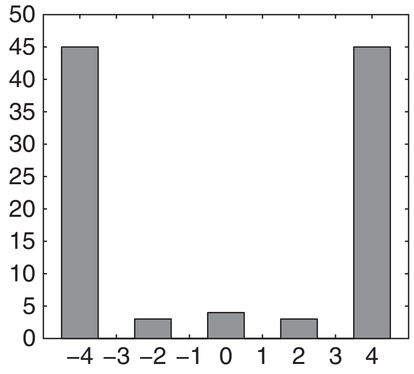
In order to illustrate and clarify these points we do the 2×2 example with free boundary conditions in detail. We assume "natural units," so that $k_B = 1$. If the temperature is high, say $T = 100$, then β is small, $\beta = 1/(k_B T) = 0.01$. Consequently, for all energies the Boltzmann factor is nearly 1. As a result, every microstate is just about as probable as any other. See columns 5 and 6 of the following table.

The picture changes markedly when the temperature is low, say $T = 1$. Then β is much bigger, with the result that the Boltzmann factor ranges through three orders of magnitude from a low of 0.018 up to a high of 54.6. In this case the low-energy microstates, with all spins parallel, are highly probable, at about 90%, while the high-energy microstates, with neighboring spins antiparallel, have vanishingly small chance of occurring.

In Figure 3.6 we show the histogram of the magnetic moment distribution for the two cases. At high temperature the distribution is somewhat bell-shaped with an equal mix of up and down spins most probable.



(a) High temperature, $T = 100$



(b) Low temperature, $T = 1$

Fig. 3.6. Magnetic moment distribution for the 2×2 Ising model.

Ising Model, 2×2 example		High Temperature vs Low Temperature					
		magnetic		$T = 100$ ($\beta = 0.01$)		$T = 1$ ($\beta = 1$)	
index	microstate	moment	E	$e^{-0.01E}$	Pr	e^{-E}	Pr
1	$\begin{matrix}++\\++\end{matrix}$	4	-4	1.04	0.065	54.6	0.45
2	$\begin{matrix}++\\+-\end{matrix}$	2	0	1	0.062	1	0.008
3	$\begin{matrix}++\\-+\end{matrix}$	2	0	1	0.062	1	0.008
4	$\begin{matrix}++\\--\end{matrix}$	0	0	1	0.062	1	0.008
5	$\begin{matrix}+-\\++\end{matrix}$	2	0	1	0.062	1	0.008
6	$\begin{matrix}+-\\+-\end{matrix}$	0	0	1	0.062	1	0.008
7	$\begin{matrix}+-\\-+\end{matrix}$	0	4	0.96	0.060	0.018	0.0001
8	$\begin{matrix}+-\\--\end{matrix}$	-2	0	1	0.062	1	0.008
9	$\begin{matrix}-+\\++\end{matrix}$	2	0	1	0.062	1	0.008
10	$\begin{matrix}-+\\+-\end{matrix}$	0	4	0.96	0.060	0.018	0.0001
11	$\begin{matrix}-+\\-+\end{matrix}$	0	0	1	0.062	1	0.008
12	$\begin{matrix}-+\\--\end{matrix}$	-2	0	1	0.062	1	0.008
13	$\begin{matrix}--\\++\end{matrix}$	0	0	1	0.062	1	0.008
14	$\begin{matrix}--\\+-\end{matrix}$	-2	0	1	0.062	1	0.008
15	$\begin{matrix}--\\-+\end{matrix}$	-2	0	1	0.062	1	0.008
16	$\begin{matrix}--\\--\end{matrix}$	-4	-4	1.04	0.065	54.6	0.45
			Z	16.004		121.22	

Microstate probabilities for a 2×2 Ising model.

However, at low temperature the distribution becomes bimodal. The most probable arrangements are all spins up or all spins down. We say that the system has undergone a *phase change*. And there is a critical temperature, called the *Curie point*, at which this takes place. One of the very interesting features of the Ising model is that it is a rather simple model but still undergoes these phase changes. This is a large part of the theoretical interest in this model.

If the temperature of the system is maintained above the Curie point, the spins lose all memory and randomize. If the temperature is then lowered too quickly, the system will lock into a pattern of islands of spin-up and spin-down domains; see Figure 3.7. These are called *metastable suboptimal states*. If, however, the temperature is lowered sufficiently slowly, the domains grow until one of the two globally minimum energy states emerges, all spins up or all spins down. This process is called *annealing*.

$$\begin{array}{cccccccccccccccc}
- & - & + & + & + & + & + & - & + & + & + & + & - & + & + & + \\
+ & + & + & + & + & - & - & + & - & - & - & - & + & - & - & + \\
+ & - & - & - & - & + & + & + & + & + & - & + & - & - & - & + \\
- & + & + & + & - & - & + & - & - & + & - & - & - & + & + & - \\
+ & - & + & + & - & - & + & - & + & - & + & + & + & + & + & - \\
+ & - & - & - & + & + & + & - & + & - & + & - & - & + & - & - \\
+ & - & + & + & - & + & + & - & + & + & + & - & + & - & - & + \\
+ & - & + & - & - & - & + & + & + & - & + & + & - & + & + & + \\
- & - & - & + & - & - & + & + & + & - & - & - & - & + & + & + \\
+ & - & - & - & - & - & - & - & + & - & - & - & - & - & + & + \\
+ & + & + & + & - & - & - & + & - & + & + & - & + & + & - & - \\
+ & - & + & + & - & - & + & + & - & - & - & + & + & + & + & + \\
+ & + & - & + & + & - & - & + & - & + & - & + & + & + & + & - \\
+ & - & - & + & - & + & + & - & - & - & + & - & + & + & + & - \\
+ & + & - & - & - & - & - & - & - & + & + & - & - & - & - & - \\
+ & + & - & + & - & - & - & - & + & - & + & + & + & + & + & -
\end{array}$$

Fig. 3.7. Metastable spin domains for a 16×16 Ising model.

```
 Matlab
> nTrials=2000; startup=3000;
> d=zeros(1,nTrials); %to histogram
> beta=.01; M=0;
> x=2*round(rand(1,4))-1; %random 4-vector of +/-1
> Hx = -(x(1)*x(2)+x(3)*x(4)+x(1)*x(3)+x(2)*x(4));
> for t = 1:startup+nTrials
>   k = fix(1+4*rand); %index to flip
>   y = x; y(k)= -x(k);
>   Hy = -(y(1)*y(2)+y(3)*y(4)+y(1)*y(3)+y(2)*y(4));
>   h = min(1,exp(-beta*(Hy-Hx)));
>   if rand <h
>     x = y;
>   end
>   if t >startup
>     Mx = x(1)+x(2)+x(3)+x(4);
      M = M + Mx; % to obtain avg mag. mom.
>     d(t-startup) = Mx;
>   end
> end
> r = -4:1:4;
> hist(d,r)
>  % compute avg moment per spin
>  M = (M/nTrials)/4;
```

3.6 Counting

The problem of counting the size of a state space Ω is closely related to the problem of sampling the space. From the computational point of view, algorithms such as we have studied for random sampling can be used with only minor modification for counting.

The link between the two is what might be called the *sheep-counting paradigm.* In earlier times, a shepherd could estimate the size of a flock by counting the number of its black sheep and applying a little arithmetic. This worked because the fraction of black sheep to white sheep remained fairly constant among flocks in a given region, and its empirical value was common knowledge among shepherds. Supposing the ratio to be $w : b$ white sheep to black sheep, then the fraction of black sheep in a flock is $s = b/(w + b)$. For example, if the ratio is 9:1, then $s = 1/10$ of the flock consists of black sheep. From this an estimate of flock size is given by

$$\text{flock size} \approx \frac{1}{s} \times \text{number of black sheep.}$$

In this section we will use this principle to estimate the number of combinatorial objects of a specified nature; results difficult to obtain in any other way.

The Monomer-Dimer Problem

Consider a material made up of atoms arranged in a regular pattern or lattice in two or three dimensions with the property that some of the atoms are chemically bonded to one of their nearest neighbors. These bonded pairs are called *dimers*; the atoms that are not bonded are the *monomers.* For a given lattice, there are many different possible arrangements of monomers and dimers that cover the lattice, called a *dimer covering.*

Most thermodynamic properties of the system can be deduced from knowledge of the number of ways of covering the lattice with given numbers of monomers and dimers. Assume that the lattice has $2m$ sites. If the covering uses s dimers, the *dimer density* is the ratio s/m. The problem is to compute the number of coverings with a fixed dimer density as the lattice size increases.

More generally, recall that in statistical mechanics, the number of possible states in a given system is given by the partition function Z. One of the features of Markov chain Monte Carlo is that the stationary distribution of the system can be approximated without knowledge of Z. But what if we want to know Z? The partition function Z gives information about the number of states with any given energy. This is a counting problem.

Generally speaking, a good strategy for solving a problem is to strip away unnecessary structure until the bare essence of the problem is revealed. With regard to combinatorial problems, the underlying nature can often be couched in graph-theoretic terms. This is the case for the monomer-dimer problem.

Consider a graph $G = (V, E)$ with vertex set V and edge set E. A set of edges $M \subset E$ is a *matching* if the edges of M are pairwise disjoint, that is, no two edges of M have a vertex in common. The vertices that occur as endpoints of the edges of M are said to be *covered* by M; the remaining vertices are *uncovered*. An edge e of M along with its endpoints constitute an abstract dimer. The uncovered vertices are the monomers. Such a matching M is one of the objects we want to count.

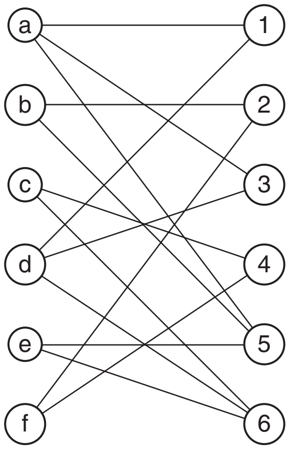


Fig. 3.8. A (bipartite) graph G with 14 edges.

An example is given by the graph with 14 edges shown in Figure 3.8. Let $\mathcal{M}$ denote the collection of matchings of G, and let $\mathcal{M}_k$ denote the collection of matchings having exactly k edges for $k = 0, 1, \ldots, 6$. This is a decomposition of $\mathcal{M}$,

$$\mathcal{M} = \bigcup_{k=0}^{6} \mathcal{M}_k.$$

The matching $\mathcal{M}_0$ has only 1 element, the empty set. We also know the size of $\mathcal{M}_1$, 14 since G has 14 edges; $\mathcal{M}_1$ will play the role of the black sheep in the sheep-counting paradigm.

It remains to devise an MCMC algorithm for generating samples from $\mathcal{M}$ uniformly. For this we need a suitable Metropolis generation/acceptance scheme, so consider the following. Let M be a given matching, and let e, with endpoints u, v, be a randomly selected edge of G. If $e \in M$, then put $M' = M \setminus \{e\}$, in other words, remove e from the current match set. If neither u nor v is an endpoint of any edge in M, then put $M' = M \cup \{e\}$. If exactly one of u and v is already covered in M, then delete the edge, say e', containing it, and again add e to the result; hence put $M' = (M \setminus \{e'\}) \cup \{e\}$. Finally, if both u and v are covered in M (by distinct edges), then do nothing, i.e., put $M' = M$. Now proceed as usual.

We first notice that since we desire the uniform distribution, we set $f = 1$; this explains why we always accept a proposed move. Secondly, notice that the above proposal scheme is symmetric, as is required.

Here is an outline of the algorithm. We will use a matrix E to define the edges of the graph and keep track of the current match set. The matrix E has a column for each edge and three rows. The first two rows are set to the end vertices of the edge; the third row is 1 or 0 depending on whether that edge is in the current match set or not.

```
Matlab
> % initialize the first two rows of E to the given graph, a 3x14 matrix
> % initialize third row to 0, start with an empty match set
> E=zeros(3,14);
> E(1,1)=1; E(2,1)=7; E(1,2)=1; E(2,2)=9;
> E(1,3)=1; E(2,3)=11; E(1,4)=2; E(2,4)=8;
> E(1,5)=2; E(2,5)=11; E(1,6)=3; E(2,6)=10;
> E(1,7)=3; E(2,7)=12; E(1,8)=4; E(2,8)=7;
> E(1,9)=4; E(2,9)=9; E(1,10)=4; E(2,10)=12;
> E(1,11)=5; E(2,11)=11; E(1,12)=5; E(2,12)=12;
> E(1,13)=6; E(2,13)=8; E(1,14)=6; E(2,14)=10;
> s=zeros(1,14); % array to count # matches of size k
> s0=0; % need to get size 0 too
> k=0; % the initial match set is empty
> for t=1:990000 % 990000 trials
    i=1+floor(14*rand); % select edge unif. at random
    if E(3,i) == 1 %check if i is in the current match set
      E(3,i)=0; % remove it
      k=k - 1; % decrement matching size
    else
      l=E(1,i); r=E(2,i); % left, right ends of i
      kl=0; kr=0; %search current match set for l and r
```

```
continued...
        for j=1:14 % loop over all edges
           if E(3,j) == 1 %this edge is in the match set
              if l == E(1,j) | l == E(2,j) % l is covered
                 kl=j;
              end
              if r == E(1,j) | r == E(2,j) % r is covered
                 kr == j;
              end
           end % if j in match set
        end %loop over edges
        if kl == 0 & kr == 0 % neither l or r covered
           E(3,i)=1; %add edge i to match
           k=k+1; % increment match set size
        elseif kl ~= 0 & kr == 0 % only left end covered
           E(3,kl)=0; E(3,i)=1; % swap edges
        elseif kl == 0 & kr ~= 0 % only right end covered
           E(3,kr)=0; E(3,i)=1; % swap edges
        end
     end % check i in current match set
>  % increment the count for the current size
>    if k == 0
        s0=s0+1;
>    else
        s(k)=s(k)+1;
>    end
  end % loop over trials
> s0 % print results
> s
```

Table 3.1 shows the results. Out of 990,000 trials the algorithm was in match state 0 a total of 2066 times, in match state 1 for 29,603 times, and so on. Since match state 0 has exactly one member, the estimate for the number of members in match state 1 is the ratio of their hits, 29,603/2066, or 14.33. In the case of match state 1 we know that the exact count is 14; so this estimate is slightly high.

To estimate the number of members in match state 2 we can use match state 0 data; this gives 152,273/2066 = 73.70. Or we can use match state 1 data and calculate $(152{,}273/29603) \times 14 = 72.01$. Continue in this way to fill out the rest of the table.

3.7 Applications

3.7.1 Shuffling with Constraints

Consider having to decide how best to play a card game such as bridge or hearts. (Knowledge of any card game will not be needed to

match size	MCMC hits (990000)	actual matches	est. using 0 (1)	est. using 1 (14)	est. using 2 (73)	est. using 3 (164)	est. using 4 (161)	est. using 5 (57)
0	2066	1						
1	29603	14	14.33					
2	152273		73.70	72.01				
3	342728		165.89	162.08	164.3			
4	335755		162.51	158.78	160.96	160.66		
5	119004		57.60	56.28	57.05	56.96	57.06	
6	8571		4.15	4.05	4.11	4.10	4.11	4.11

Table 3.1. MCMC hits and resulting match size estimates.

understand the problem in this application.) The difficulty is in not knowing what cards the other players have. However, as play continues, partial information is revealed about those cards. Thus one reasonable plan is to generate a strategy that is stochastically good. That is, one that performs well given a random allocation of the unknown cards to the other players.

To pose a very specific problem, assume that there are four players, counting the hand we play, and at a certain point in the game, the disposition of all cards is known except for these 18:

$$\begin{aligned}&\text{club: } 3, 4;\ \text{diamond: } 7, 8;\ \text{heart: } 2, 3, 5, 6, 7, 8, 9;\\ &\text{and spade: } 2, 3, 4, 5, 6, 9, \text{queen.}\end{aligned} \tag{3.24}$$

Referring to the other three players as hands 1, 2, and 3, each of them has six of these cards, but it is not known which six. However, there is additional information: it is also known that hand 1 has no clubs and hand 2 has neither clubs nor diamonds.

The problem is to generate a random allocation of the unknown cards consistent with the suit constraints. In other words, we want to generate random shuffles subject to constraints. Given such a legal allocation, the game would be one of perfect information, like chess or checkers, and allow the computation of the best move at any time. Since any given legal allocation is not necessarily the actual allocation (that remains unknown), the strategies devised have to be of a minimax or stochastically optimal nature.

One legal allocation of the mystery cards above is the following: hand 1: heart 2, 3, 4, 5, 6, 7; hand 2: spade 2, 3, 4, 5, 6, 9; and hand 3 has the rest. This seems hardly random, but it can be used as the starting point for an MCMC solution. Since we want a uniform distribution over all legal permutations x of the mystery cards, our desired Metropolis density is $f(x) = 1$ for all x; hence the acceptance discipline is to accept with certainty, that is, always accept.

It remains to create a symmetric proposal function fostering an irreducible chain. The algorithm is:

1. Start with a legal allocation of the unseen cards.
2. Select a card, say A, from hand 1, a card B from hand 2, and a card C from hand 3.
3. Attempt to swap A and B, or A and C, or B and C, or A to B to C, or A to C to B (in random order).
4. For the first of these to produce a legal allocation, accept it and continue; if none do, keep the original allocation and continue.

This scheme is able to convert any legal shuffle to any other. The invariant distribution of this chain is the uniform distribution over all legal allocations. As with any Markov chain simulation, enough iterations must be performed in order for the invariant distribution to be approximated.

3.7.2 Coupling from the Past

If a Markov chain could be run for infinite time, then the state random variable X_t would be an exact sample from the invariant distribution. What amounts to the same thing, if the chain was started in the infinite past, then the state X_0 at time 0 is a sample from the invariant distribution exactly. Remarkably, it is possible to achieve this. Here is how.

First select a finite set of random numbers just as you would to simulate the chain, say 10 such numbers, call them $\{r_{-1}, r_{-2}, \ldots, r_{-10}\}$. Now start at $t = -1$, and for each state of the chain, take one step forward in time using the same "random number" r_{-1}. If taking one step forward in time from each state results in one and only one state at time $t = 0$, then stop and report that state. Otherwise, repeat this from time $t = -2$ but always using the originally selected random numbers.

As a specific example, here is how it would go using the stock market chain (from Section 3.1). Recall the states are U for up, N for no change, and D for down, and the transition matrix is

$$P = \begin{bmatrix} 0.3 & 0.2 & 0.5 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}.$$

The corresponding cumulative matrix is

$$C = \begin{bmatrix} 0.3 & 0.5 & 1.0 \\ 0.4 & 0.6 & 1.0 \\ 0.4 & 0.7 & 1.0 \end{bmatrix}.$$

Assume that the starting random numbers are $\{0.31, 0.57, 0.97, 0.85\}$. Beginning with $t = -1$ and the random number 0.31, if the state is U, then the transition is to N; if the state is N, the transition is to U; and if the state is D, the transition is to U as well; see Figure 3.9. This leaves the chain in two possible states, U and N, and so the chain has not "coupled"; we must go back to $t = -2$ and use the random number 0.57.

For a random number 0.57, the transition from U is to D, from N to N and from D to N. Now the states have coupled at $t = 0$. Coming from the infinite past up to time $t = -2$ could put the chain in any of U, N, or D for all we know. But the next transition leaves the chain in either N or D at $t = -1$. Then in the next transition these two states move to U. So the chain is in only one state at $t = 0$, namely U. This instance of the random numbers r_{-1}, r_{-2}, ... leaves the chain in state U at $t = 0$.

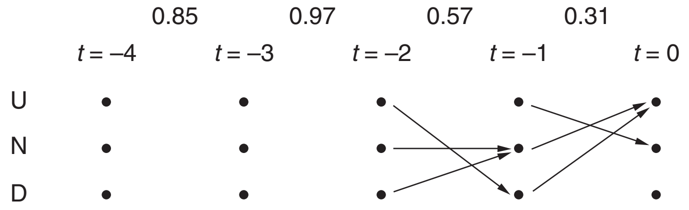


Fig. 3.9. Coupling from the past for the stock market chain.

To produce another sample from the invariant distribution, the procedure is started all over again by making another selection of random numbers r_{-1}, r_{-2}, ...

Programming coupling from the past is somewhat tricky, because if the chain does not couple using the initial selection of random numbers $r_{-1}, r_{-2}, \ldots, r_{-10}$, then one must go back further, select $r_{-11}, r_{-12}, \ldots$, r_{-20}, and see whether coupling occurs from 20 steps back. Meanwhile, the initial selection of random numbers must be saved, since they will be needed to follow the chain from repetition -10 to 0.

The following example shows that one cannot stop as soon as coupling occurs, as in this case this would always result in the same value being returned. Consider the chain with transition matrix

$$P' = \begin{bmatrix} 0 & 1 & 0 \\ 1/3 & 1/3 & 1/3 \\ 0 & 1 & 0 \end{bmatrix}.$$

The invariant distribution is $[\,1/5 \quad 3/5 \quad 1/5\,]$. Note that the only coupling possible is in state 2; see Figure 3.10. Suppose the initial set of random numbers is $\{0.86, 0.39, 0.85, 0.06, 0.20\}$. Using 0.86 at time -1 to time 0, we get

$$1 \to 2$$
$$2 \to 3$$
$$3 \to 2$$

so 1 and 3 have coupled but 2 has not. Starting from time -2 and using the random value 0.39 to transition to time -1 we get

$$1 \to 2$$
$$2 \to 2$$
$$3 \to 2$$

so all have coupled in state 2. Now using the random value 0.86 as before to go to time 0, we have that $2 \to 3$ as before, so 3 is the result of this trial and is a sample from the invariant distribution.

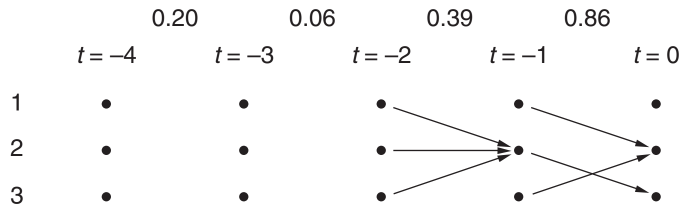


Fig. 3.10. Coupling always happens in state 2 for chain P'.

The following example shows that one cannot stop after a fixed number of iterations and try again if coupling has not yet occurred. Consider the transition matrix

$$P'' = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \tag{3.25}$$

Again it is possible to couple only in state 4 because states 1 and 7 go only to state 4. Further, once coupling has taken place, it requires at least three iterations to reach either state 1 or 7, usually more. Now assume that we go back in time a fixed number of iterations, for example 6, and restart if coupling has not taken place by then. The consequence is that one will not observe states 1 and 7 as frequently as they should appear.

Notes

A good source for more information on Markov chains is the book [Bre99]. The theory presented here concentrates on finite-state-space Markov chains. This is sufficient for practical computations, since any system implemented on a computer will necessarily be finite. However, for modeling this is not enough. Much of the theory can be extended to countable state spaces with little change. Continuous state spaces are more difficult, as are continuous-time Markov processes.

The basic Metropolis algorithm was greatly generalized by Hastings [Has70], so it is often called the Metropolis–Hastings algorithm. There is a wide range of MCMC algorithms suited to many different applications (for a place to start, see [GH05, HH64, KW86, Liu01, Rub81, Yak77]). In particular, statistics has recently become a very large user of MCMC.

It is often important to have some idea of the convergence rate of a Markov chain to its stationary distribution. Similarly, some idea of the decay of the error in an MCMC estimation is also very useful, or statistical estimates such as confidence intervals might be needed. In general, estimating the finite-time behavior of a Markov chain is a difficult task. Most of the references above contain some discussion on these topics.

The technique of exact sampling using couping from the past is particularly useful for special types of chains. One case is when there is a natural partial order on the state space and the Markov chain evolution preserves this order (see [PW96]). In this case, all that is necessary is to check whether the "top" and "bottom" states couple since then all the in-between states will have coupled as well. Some versions of the Ising model satisfy this monotone property and thus the coupling from the past technique may be used on them.

Problems: Chapter 3

1. (4) Suppose the state space for a system of particles is $\Omega = \{1, \ldots, 8\}$ and their allocation among these states is determined by the Boltzmann distribution. Let the energy of state x be $E(x) = (x-3)^2$. Use the Metropolis algorithm, with $\beta = 1$ (see (3.21)), to estimate the expected "blackbody radiation" of the system if the radiation r is equal to the fourth power of energy, that is, take $r(E) = E^4$. (Exact: 6.166.)

2. (5) With the setup as in the problem above, take the neighborhood system to be uniform over the space of eight states. (Or pick some other if you like.) Now compute the transition matrix P for Metropolis sampling. Find the second-largest eigenvalue μ of P. The "spectral gap" is defined to be $1 - |\mu|$. It gives the geometric rate for how fast the starting distribution p_0 converges to the stationary distribution π (the Boltzmann distribution).

3. (3) In this problem we compute the probability distribution p_t for the stock market example, but using various starting distributions p_0. The case $p_0 = [\,1 \quad 0 \quad 0\,]$ is already done in the text. For this problem, do the case $p_0 = [\,0 \quad 0 \quad 1\,]$. Compute p_t for $t = 1, 2, \ldots, 10$ for both choices of p_0. Using these computations, compute p_t starting with a general $p_0 = [\,a \quad b \quad 1-a-b\,]$ where $a, b \geq 0$ and $a + b \leq 1$ (use the fact that matrix multiplication is linear). Observe that for any starting p_0, the distributions p_t seem to converge to the invariant distribution.

4. (4) Sample from the distribution on $\Omega = \{2, 3, \ldots, 12\}$ that one obtains by rolling two fair six-sided dice by using the Metropolis algorithm. First, use the uniform distribution over Ω as the neighborhood system. Next use the random walk neighborhood system given by the matrix in equation (3.8). Record the computation time and make a histogram and correlation plot. Comment on your observations.

5. (5) This problem experimentally examines the rate of convergence of an MCMC scheme. Sample from the distribution obtained by rolling two fair six-sided dice using the Metropolis algorithm with neighborhood system as given by the matrix in (3.8). Letting the sequence of samples be $X_1, X_2, \ldots, X_N$, compute

$$\text{avg}_N = \frac{1}{N} \sum_i X_i.$$

Run 100 instances of $N = 100$, histogram the values of avg_{100}, and compute the variance. Repeat with 100 instances of $N = 200$ and again with 100 instances of $N = 1000$. This should give an indication of how the estimates of the average converge as the number of iterations increases.

Now do this all again except with the maximal neighborhood system in which all states are chosen with equally likelihood. Compare the two different neighborhood systems in terms of their convergence properties (spread of histogram and variance).

6. (4) Use Metropolis to sample from the pdf $f(x) = cx^2e^{-x}$, $0 \le x < \infty$. As the neighborhood of x, take the interval $[x-\delta, x+\delta]$ for some $\delta > 0$. (Near $x = 0$ reject negative proposals; see the text.) Check pdf, correlation, efficiency, effect of start point (by doing a pdf starting from different points).

7. (5) Obtain a shuffle of a 52-card deck by the repetition of an elementary shuffle operation (ESO). Create your own ESO. Some examples are "divide the cards into two equal piles and interleave," "a random two-card swap," and "slide the top card into a randomly selected location." Repeat the ESO enough times until the cards are completely randomized and return the result. That is to say, obtain the uniform distribution over all permutations of the cards so all permutations are equally likely. This idea would be an instance of Markov chain Monte Carlo, each ESO being the "generation" step; since the target distribution is the uniform one, all proposals are accepted. Test the resultant shuffles in some way. For instance, for runs up or streaks of even numbers or correlation or quartile mixing. How many ESOs are needed to randomize the cards? (Plot your test as a function of the number of ESOs.)

8. (7) (Shuffle under constraints.) Obtain random shuffles of the cards: club $2, 3, 4, 5, 6$; diamond $2, 3, 4, 5, 6$; heart $2, 3, 4, 5, 6$; and spade $2, 3, 4$; in such a way that no clubs or spades appear in positions $1, 4, 7, \ldots$, no hearts appear in positions $2, 5, 8, \ldots$, and no diamonds or spades appear in positions $3, 6, 9, \ldots$.

9. (3) (Coupling from the past) Use the coupling from the past method to obtain samples from the invariant distribution of the stock market chain. Compare with the actual distribution $\left[\frac{37}{112}, \frac{34}{112}, \frac{41}{112}\right]$.

10. (4) (Coupling from the past) Use the coupling from the past method to obtain samples from the invariant distribution for the chain defined by P'', equation (3.25) on page 134. What happens if you restart after some fixed number, N, of iterations if coupling hasn't occurred? Try with $N = 6, 7, 8$ and see how this introduces a bias in the computed "invariant distribution". For this matrix P'', the actual invariant distribution is $\left[\frac{1}{26}, \frac{1}{13}, \frac{2}{13}, \frac{6}{13}, \frac{2}{13}, \frac{1}{13}, \frac{1}{26}\right]$.

11. (5) (Noncomputational) Let P be the transition probability matrix of a chain for which detailed balance holds. Show that all the eigenvalues of P

are real. (Hint: letting π denote the invariant distribution and Π the diagonal matrix whose diagonal is π, show that $\Pi^{1/2}P\Pi^{-1/2}$ is symmetric.)

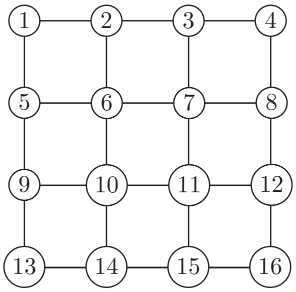


Fig. 3.11. Monomer-dimer problem.

12. (4) (Monomer-dimer problem) In Figure 3.11 we show a 4×4 square two-dimensional lattice as a graph. (a) Find the number of dimer coverings of different sizes for this lattice. (b) Do the same for a 3×3 and a 5×5 lattice. Plot the number of dimer covers for various dimer densities as a function of the lattice size.

13. (5) Simulate the Markov chain with transition matrix

$$P = \begin{bmatrix} c_0 & \frac{1}{2}e^{-3} & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2}e^{-1} & c_2 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2}e^{-1} & c_3 \end{bmatrix},$$

where the c_i's are the complementary probabilities (so the rows sum to 1) for a few hundred iterations to obtain the invariant distribution and show the histogram. Calculate the invariant distribution by hand. Compare with the following calculation: $w_i = e^{-f_i}/Z$ where $Z = \sum_{i=0}^{3} e^{-f_i}$ and $f_0 = -1$, $f_1 = 2$, $f_2 = 1$, $f_3 = 0$.

14. (6) In this problem, we investigate exit times from a set of states. With P as in the problem above, let N be the set of states corresponding to the last three rows of P. Then the matrix $\hat{P}$ consisting of the lower right 3×3 block,

$$\hat{P} = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2}e^{-1} & c_2 & \frac{1}{2} \\ 0 & \frac{1}{2}e^{-1} & c_3 \end{bmatrix},$$

is the transition matrix among those states. (a) By starting the chain in N, estimate the expected number T of iterations to exit N and the probability

p_N that the chain exits N per iteration. What is the relationship between T and p_N?
(b) Calculate the largest eigenvalue λ of $\hat{P}$. What is the relationship between λ and p_N?

15. (8) (Particles in a 2D box) Assume that three particles are in the box $0 \leq x \leq 1$, $0 \leq y \leq 1$. The state $\mathbf{x} \in \Omega$ is the vector of positions and velocities of each particle,

$$\mathbf{x} = \begin{bmatrix} x^1 & y^1 & v_x^1 & v_y^1 & x^2 & y^2 & v_x^2 & v_y^2 & x^3 & y^3 & v_x^3 & v_y^3 \end{bmatrix},$$

where (x^i, y^i) is the position of the ith particle and (v_x^i, v_y^i) is its velocity vector. The energy $E(\mathbf{x})$ of the system equals

$$E = \frac{1}{2}\sum_{i=1}^{3} m^i \left((v_x^i)^2 + (v_y^i)^2\right) + \sum_{i<j} \Phi(r_{ij}),$$

where m^i is the mass of the ith particle (take m^i to be 1 for all i), $|v^i|$ is its speed, r_{ij} is the distance between the ith and jth particles and $\Phi(\cdot)$ is the Lenard-Jones potential

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

Here ϵ and σ are physical constants; take $\epsilon = 1$, and since σ acts like the radius of a particle, take it to be 0.1.
For the proposal, or perturbation, scheme, select a particle equally likely and perturb its x and y components of velocity, Δv_x and Δv_y, according to a Gaussian with mean 0 and variance 1. Move all particles according to their velocities for an increment in time of $\Delta t = 0.05$ second (reflect off the walls if necessary). This is the proposed new state to be decided for acceptance or not according to the Metropolis algorithm for the Boltzmann distribution; see (3.23). Try to find a temperature T such that around 80% of the perturbations are accepted. Histogram the average speeds of the particles over the course of the simulation.

4 Optimization by Monte Carlo Methods

It may be that the problem of optimization entails more time and effort by mankind than any other mathematical problem. For example, it permeates nearly all design and engineering projects.

When formalized, the problem consists of a given function f called the *objective function* together with a domain Ω of its arguments x. It is required to find a value of $x \in \Omega$ that minimizes or maximizes $f(x)$. The problem may also impose certain constraints on the *admissible* or *feasible* x's. In the formalization above, Ω would (usually) exclude infeasible states. However, it is often better to leave infeasible states in the domain, since they can be useful in searching for a feasible solution. In fact, one method of attacking a constrained optimization problem is to penalize infeasible states, and to an increasing degree as the run progresses.

Among the more difficult optimization problems are those of a combinatorial nature. Some of these are quite famous, such as the traveling salesman problem (TSP). There are ongoing attempts to solve such problems by deterministic methods. A solution would produce a *provably* global minimizer (or maximizer) in a reasonable amount of time. The traveling salesman problem has seen the most effort along these lines. Unfortunately, as of the present date no general solution is yet available. Deterministic algorithms that do exist for combinatorial problems must leverage deep insights into whatever theory underlies the problem, for example Euclidean geometry. As a result, these algorithms are quite complex.

A second category of especially problematical optimization problems consists of those having numerous local minima (maxima), possibly numbering in the millions. Even given an efficient method for finding local optima, how can that be used to find global optima?

As a result of these difficulties, a Monte Carlo method is often the method of choice for attacking an optimization problem. Monte Carlo methods are usually very simple to program, and they locate good, if not the absolute best, solutions in reasonable time. The two most widely known and used Monte Carlo optimization methods are simulated annealing (SA) and genetic algorithms (GA).

4.1 Simulated Annealing

One of the most successful techniques for solving such problems is the adaptation of ideas from statistical mechanics. As we have seen in Section 3.5, thermal systems can be annealed to achieve minimal internal energy states. This is brought about by raising the system to a high temperature and then cooling it slowly. Since thermal systems can be computationally simulated, it is natural to imagine that a similar type of annealing could be performed on a more general optimization problem, leading to the determination of the global minimum of the objective function. The method is called *simulated annealing* (SA) and was first proposed in 1983 (see [KGV83]).

Just as in the simulation of thermal systems, the central component of simulated annealing is the Metropolis algorithm; see Section 3.3. The domain Ω serves as the system's phase space, and the objective function serves as its energy. As such, the optimization problem must be cast as a minimization problem. To this end, using the objective function as the energy, the probability of accepting a proposed move from x to y is given as

$$h = \min(1, e^{-(f(y)-f(x))/k_B T}). \tag{4.1}$$

In the same way as in a statistical mechanics simulation, the Metropolis algorithm imposes a Boltzmann distribution on the sampled microstates x (which we now refer to as *solutions*) so that the probability density is

$$\frac{e^{-f(x)/k_B T}}{Z}. \tag{4.2}$$

From (4.2) it is clear that for T close to zero, the points that minimize f will dominate the probability distribution.

The remaining ingredient is a proposal mechanism to induce a shuffling through Ω, the space of solutions. As before, the proposal mechanism can be thought of as an $|\Omega| \times |\Omega|$ matrix g where g_{xy} is the probability of proposing y from x. Again g should fulfill two requirements, the same two as for the Metropolis algorithm. The first is that it be *irreducible*, meaning that for any x and y in Ω, there is a finite sequence $x_0 = x$, x_1, …, $x_n = y$ such that $g_{x_{i-1}x_i} > 0$ for $i = 1, \ldots, n$. Secondly, g should be symmetric: $g_{xy} = g_{yx}$ for any two solutions x and y, which is to say that x is proposed from y with the same probability that y is proposed from x.

Cooling schedules

The other central feature of simulated annealing for global optimization is the imposition of an abstract temperature. The temperature must be lowered in some fashion over the course of a run. The exact prescription for how this is done is called the *cooling schedule*. The choice of cooling schedule is critical to the performance of a simulated annealing algorithm. As can be seen from (4.2), at low temperatures, solutions with high energy have very low probability under the Boltzmann distribution. Because of the form of the acceptance (given in (4.1)), at low temperatures the system tends to move "downhill" only (in terms of objective function values). This is good for finding the local minimum of the current basin; however, the system also tends to lodge at the local minimum. At high temperatures energy differences hardly matter, and the system can easily escape shallow basins in order to explore the energy landscape more widely, but it is almost as likely to find high-energy states as it is to find low-energy states. The rough idea behind cooling is to start with high temperatures to identify "good regions" of the state space and then to explore these regions fully in the lower temperature phase. It is important to cool slowly enough so that it is always possible to reverse any "mistakes" made at higher temperatures.

It should be noted that because the temperature is changing at each iteration, the acceptance scheme (4.1) will cause the Markov chain to have time-varying transition probabilities. Such a chain is called an *inhomogeneous Markov chain*. The analysis of inhomogeneous chains is much more difficult than that for time-invariant chains.

Exactly how to cool has been worked out and is specified by Hajek's theorem [Haj88]. Let G denote the set of global minimizers in Ω. Among other assertions, the theorem provides the following.

Theorem 1 *Under an irreducible and symmetric proposal, the probability that the Markov chain X_t will be in a globally minimal state tends to certainty as $t \to \infty$,*

$$\lim_{t\to\infty} \Pr(X_t \in G) = 1,$$

if temperature as a function of iteration t is taken as

$$T_t = \frac{c}{\ln(1+t)}, \qquad t = 1, 2, 3, \ldots, \tag{4.3}$$

provided the constant c is sufficiently large.

Cooling as specified by (4.3) is called *inverse log cooling* (or sometimes *log cooling*). The constant c should exceed the maximum of the depth of all basins containing only local minima. The precise definition of the depth of a basin is somewhat involved, and in almost all cases these depths cannot be known or computed in advance, only estimated. For our purposes, therefore, the intuitive notion will suffice.

Another shortcoming of the theorem with respect to applications is the requirement that t tend to infinity. Since an actual anneal cannot run for infinite time, the choice of starting temperature is of some importance. Furthermore, under these conditions, inverse log cooling may not be the best choice. Rather *geometric cooling*, namely

$$T_t = ab^t, \qquad a > 0, \quad 0 < b < 1, \tag{4.4}$$

for some parameters a and b, can be a better choice. It is often necessary to do some preliminary runs to decide between inverse log and geometric cooling, the starting temperature, and the cooling parameters a, b, and c.

As in real thermal systems, a simulated anneal often experiences a phase change. This is characterized by the system's energy decreasing rapidly at some temperature. After this, the system makes improvements much more slowly. The choice of cooling schedule should bracket any phase-change temperatures for the problem, if at all possible, for best performance.

The best way to get a feeling for simulated annealing is to see it in action.

4.2 Application of SA to the Traveling Salesman Problem

Given n cities in terms of their geographical coordinates C_i, $i = 1, \ldots, n$, a *tour* of the cities is a path starting at C_1 and leading from one city to the next in some order and finally back to C_1. Each city is to be visited exactly once (other than C_1). Evidently a tour is uniquely identified by a permutation x of the set $\{2, 3, \ldots, n\}$. For example, if $x = (i_1, i_2, \ldots, i_{n-1})$ is such a permutation, and we let $i_0 = 1$, the corresponding tour is

$$C_{i_0}, C_{i_1}, \ldots, C_{i_{n-1}}, C_{i_0}.$$

Let Ω be the set of all such tours.

The *length of a tour* is the length of its path. For the tour above, this is

$$E(x) = \sum_{k=1}^{n-1} d(C_{i_{k-1}}, C_{i_k}) + d(C_{i_{n-1}}, C_{i_0}),$$

where $d(C_i, C_j)$ is the distance from city C_i to C_j. Take this length as the "energy" or objective $E(x)$ of $x \in \Omega$.

The *traveling salesman problem* (TSP) is that of finding the tour having the smallest length. Of course, the reverse of a given tour has the same length and for our purposes, can count as the same tour.

For n cities there are $(n-1)!$ permutations of the set $\{2, 3, \ldots, n\}$, and half that when accounting for reverse tours; this can be a large number for even small values of n. Hence finding the minimal tour is a very hard problem.

The TSP exhibits several features that make it a good problem to use as a testbed for an optimization method. In particular, the fact that the problem becomes exponentially harder as the number of cities increases is particularly useful, as is the fact that there are many variations for each problem size (number of cities).

Partial Path Reversal

To attempt a solution by simulated annealing, a proposal scheme must be defined, that is, a perturbation for any given tour. One such proposal scheme is *partial path reversal* (PPR). Let $x = (i_k : k = 2, \ldots, n)$ be a tour and randomly select two terms of the sequence, say i_{k_1} and i_{k_2}. Now reverse the part of the tour between and including these two cities.

For example, suppose 3, 5, 2, 8, 4, 6, 7 is a tour for an 8-city TSP and suppose the terms $i_{k_1} = 5$ and $i_{k_2} = 4$ are selected. Then the resulting PPR tour is 3, 4, 8, 2, 5, 6, 7. A visual representation of this is

$$3, \mathbf{4}, \mathbf{2}, \mathbf{8}, \mathbf{5}, 6, 7$$
$$\text{changes to}$$
$$3, \mathbf{5}, \mathbf{8}, \mathbf{2}, \mathbf{4}, 6, 7.$$

A Simulated Annealing Algorithm for the TSP
In the following algorithm we save the best energy B encountered during the run.

```
generate a random permutation $x$ of $\{2, \dots, n\}$ and calculate $E(x)$.
$T = T_0$, $B = E(x)$
loop $t = 1, 2, \dots$
   generate a random PPR $x'$ of $x$ and calculate $E(x')$.
   $\Delta E = E(x') - E(x)$, if $\Delta E \le 0$ replace $x$ by $x'$ (and $E(x)$ by $E(x')$)
   otherwise put $h = e^{-\Delta E/T}$ and generate a
      uniformly distributed random number $0 \le U < 1$
   if $U < h$ replace $x$ by $x'$ (and $E(x)$ by $E(x')$)
   put $B = \min(B, E(x))$
   update the temperature T=T(t)
end loop
```

Fig. 4.1. Traveling salesman problem annealer using partial path reversal.

The general outline of a simulated anneal for the TSP is given in Figure 4.1. It incorporates Metropolis acceptance/rejection and PPR. This algorithm was run on the Bays29 problem consisting of 29 Bavarian cities, see Figure 4.2 and the notes at the end of this chapter.

It remains only to specify the cooling schedule. The cooling schedule as spelled out in Hajek's theorem pertains when the number of iterations is unbounded. But setting the schedule in an actual problem is more of an art. As mentioned above, it is often observed while monitoring the progress of a run that the algorithm makes especially dramatic improvement at some point during the run. This is known as a "phase change" in analogy to real annealing. A cooling schedule should ensure that the phase change temperature is bracketed.

We consider a heuristic for setting the cooling schedule. Start by deciding the number of iterations to be run; we will take that to be 10^5 for Bays29. Next estimate the average absolute value of ΔE for the

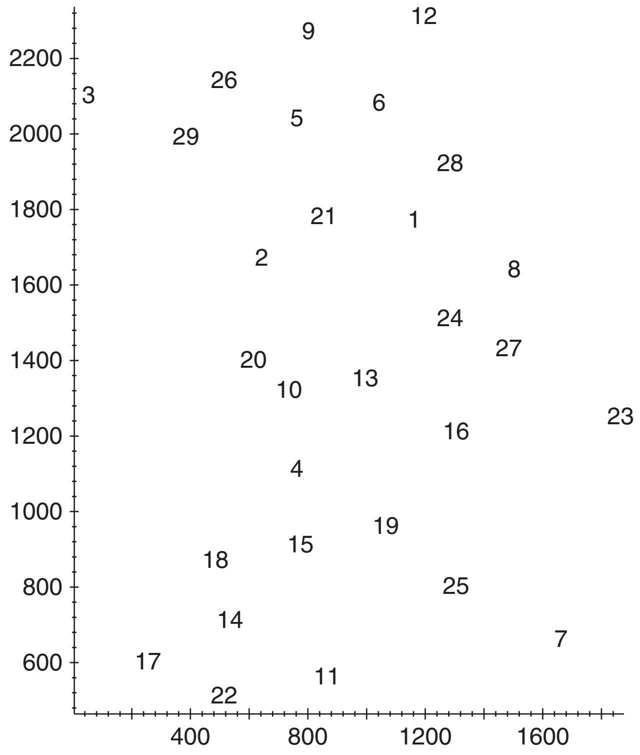


Fig. 4.2. Spatial location of the cities of the Bays29 problem.

random perturbation operation chosen, for example, PPR in the traveling salesman problem. This is easy to do by performing a small number of preliminary iterations of the algorithm and just noting the changes in energy. This came out to be about 1000 for Bays29. Finally, decide on the form of the cooling schedule; we will take inverse logarithmic cooling according to the equation

$$T = \frac{c}{a + \log t}$$

with c and a parameters to be determined.

Now arrange it so that the probability of accepting an average uphill ΔE at the beginning of the run is near 1 and at the end of the run is near 0. We chose 0.99 and 10^{-5} respectively. Thus the ending temperature T_∞ is given by

$$10^{-5} = e^{-1000/T_\infty}, \quad \text{so} \quad T_\infty = 86.86,$$

and therefore c is given by

$$86.86 = \frac{c}{\log 10^5}, \qquad c \approx 1000.$$

Note that a will be very small compared to $\log 10^5$ and can be neglected here. With c in hand, a can be determined. The starting temperature is given by

$$0.99 = e^{-1000/T_0}, \qquad T_0 \approx 99,500.$$

Since $\log(1) = 0$, we get a from

$$99500 = \frac{1000}{a}, \qquad a \approx 0.01.$$

In Figure 4.3 we show the progress of the above algorithm for the Bays29 problem. The envelope of best and worst of 10 runs is shown, that is, at each iteration count, the best and the worst out of the 10 runs are plotted. Notice how most of the progress is made in the first part of the run, with very little improvement after the first 25,000 iterations or so. This behavior is quite typical for SA.

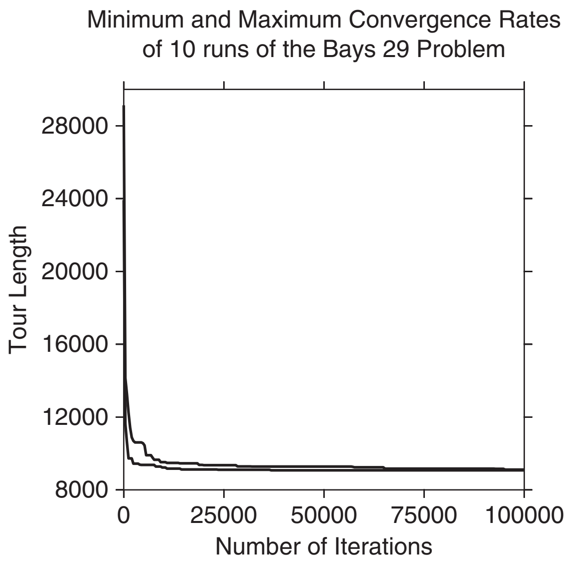


Fig. 4.3. Envelope of the best and worst of 10 runs of the anneal to solve Bays29.

4.3 Genetic Algorithms

The method of genetic algorithms is an optimization method based on the paradigm of biological adaptation (see [Gol89]). The essential ingredients of the method are mechanisms for recombination, mutation, and selective reproduction working on a population of potential solutions. Fitness for a solution is directly related to the objective function being optimized and is greater for solutions closer to its global maximum (or minimum). The idea is that by repeated application of the genetic and selection operations, the population will tend toward increased fitness. Using a population of potential solutions, rather than a single potential solution as in SA, allows a genetic algorithm search to recover from incorrect moves.

A Genetic Algorithm Is a Markov Chain

A genetic algorithm generates a Markov chain X_t on *populations* over Ω under the action of the three stochastic operators *mutation*, *recombination*, and *selection* defined on Ω. Although implementation details may vary, mutation is a unary operator, recombination or *crossover* is a binary operator, and selection is a multiargument operator. Under any reasonable definition of the three operations, the chain X_t is always irreducible and aperiodic and so converges to a (unique) stationary distribution. Two example algorithms are given in Figures 4.4 and 4.5. In addition, an implementation of Algorithm 1 in Matlab is given in Section 5.5.

GA Algorithm 1

```
initialize population $C_0$
initialize $B$ to the best fitness of $C_0$
loop $t = 1, 2, \dots, N$
   use roulette-wheel selection to create population $C_1$
   crossover chromosomes 1 with 2, 3 with 4, ...
      with each pair of offspring replacing their parents
   loop over every chromosome $c$ in $C_1$
      with probability $p_m$, mutate $c$
   end loop over chromosomes
   $C_0$ becomes $C_1$ for the next iteration
   update the best fitness $B$
end loop over generation $t$
```

Fig. 4.4. A complete population turnover style GA.

The implementation of a genetic algorithm begins with the computer representation, or encoding, of the points x of the solution space Ω. Frequently this takes the form of fixed-length binary strings, which are called *chromosomes*. A natural mutation of such a string is to reverse, or flip, one or more of its randomly selected bits. Likewise, a natural recombination of two bit strings, called *parents*, is to construct a new binary string from the bits of the parents in some random way. The most widely used technique for this is *one-point crossover*, in which the initial sequence of k bits of one parent is concatenated with the bits beyond the kth position of the second parent to produce an *offspring*. Here k is randomly chosen. A visual representation of this is:

$$\begin{array}{ll} \text{parent 1} & \overbrace{10110101}\,010100 \\ \text{parent 2} & 10111010\,\underbrace{110010} \\ \text{offspring} & 10110101\;110010 \end{array}$$

Of course, a fitness evaluation must be done for each new chromosome produced.

GA Algorithm 2

```
allocate popSize times 3 space for the population
initialize a popSize population $C$ of chromosomes (ch.)
sort the population according to fitness (best to worst)
initialize $B$ to the best fitness of $C$ (the first ch.)
loop $t = 1, 2, \ldots, N$
   loop $i = 1, \ldots,$ popSize/4
      select a random ch., copy it, mutate the copy and merge
      the copy into the population according to its fitness
   end loop
   loop $i = 1, \ldots,$ popSize/2
      crossover ch. of rank $i$ with a randomly selected ch.
      in the population and merge the offspring(s) into the pop.
   end loop
   *Comment: the pop. is now about doubled in size and in sorted order*
   update the best fitness $B$
   loop until the population is down to popSize
      discard one member of the population by Russian
      roulette with probability 1/6, starting with
      the worst and working toward the best,
   end loop
end loop over generations $t$
```

Fig. 4.5. A sorted rank style GA.

Finally, the chromosomes selected to constitute the population in the next generation might, for example, be chosen by lottery with the probability of selection weighted according to the chromosome's fitness. This widely used method is termed *roulette wheel selection.*

Most genetic algorithms do not vary the parameters of their three operators over the course of a run, and therefore, in contrast to simulated annealing, the transition probabilities are constant throughout. Such a Markov chain is said to be *homogeneous.* One consequence is that the algorithm is unlikely to be in the globally optimal state at the end of a run, since the chain is irreducible and will visit all states recurrently. Therefore it is important to save the best value discovered by the algorithm, and the corresponding structure, from iteration to iteration.

Niching and Remedial Measures

As the run of a genetic algorithm progresses, improvements in fitness begin to occur more and more slowly. Eventually the algorithm fixates or *niches* at a local maximum (a global maximum perhaps?). This niching typically occurs when all or most of the current population becomes copies (or near copies) of one individual. But by construction, the algorithm is an irreducible Markov chain, and therefore, given enough time, the process will move on and find another local maximum. This is even true of the global maximum.

It could take a very long time for a transition to occur, and so it can be better to stop the run manually and restart the process afresh. In this way, a different part of the landscape will be searched with the possibility that the new local maximum uncovered will be also global.

A modification of the algorithms, called wrong-side-of-the-tracks crossover, induces a much greater chance of transition out of niched states. After the algorithm makes no improvement for a period of time, crossover is modified to take place between one member of the current population and another member chosen at random from the entire state space Ω. This induces a *tempered* as opposed to a hard restart.

4.4 An Application of GA to Function Maximization

For our first illustration of genetic algorithms, we try to find the maximum of a given function f defined on an interval of real numbers. A *Shekel function* has the form

$$f(x) = \sum_{i=1}^{n} \frac{1}{a_i(x-b_i)^2 + c_i}, \qquad a_i c_i > 0. \tag{4.5}$$

If the a_i and c_i are positive, then $f(x) > 0$ for all x. The ith term of the sum achieves a maximum value of $1/c_i$ when $x = b_i$. Otherwise, the term decays to zero more rapidly the larger the value of a_i. Thus f is a series of peaks located approximately at the b_i, since each peak is affected to a greater or lesser extent by the others depending on their proximities and decay rates.

By taking the a_i and c_i all negative, f is reflected across the x-axis. The peaks now become pits, and a search for minima is in order.

The Shekel functions may easily be extended to Euclidean spaces of any dimension, for example in two dimensions they take the form

$$f(x,y) = \sum_{i=1}^{n} \frac{1}{a_{i1}(x-b_{i1})^2 + a_{i2}(y-b_{i2})^2 + c_i}.$$

Now the peaks are at the points (b_{i1}, b_{i2}), $i = 1, \ldots, n$, and the x-directed decay rate is a_{i1}, while the y-directed rate is a_{i2}.

a	b	c	a	b	c
10.295	7.382	0.1613	6.163	9.517	0.2204
4.722	1.972	0.2327	13.860	8.571	0.0838
10.928	8.111	0.2047	5.310	6.918	0.0665
11.052	2.714	0.1579	4.047	1.618	0.1719
3.820	4.022	0.1268	8.085	7.771	0.1040
12.911	5.480	0.0519	7.974	8.316	0.1340

Table 4.1. Parameters for a 12-term Shekel Function.

The function we will try to maximize is the 12-term Shekel function with parameters as in Table 4.1 over the interval $0 \le x \le 10$. Its graph is shown in Figure 4.6. We will illustrate with two somewhat different algorithms, given previously in Figures 4.4 and 4.5. In both, points x in the interval are coded as 10-bit binary numbers; denote such a 10-bit string by c. Thus c is an integer from 0 to $2^{10} - 1 = 1023$. The corresponding x is given by $x = 10c/1023$. We use a population size of 12 in each algorithm.

Figure 4.7 shows the solution history of six runs of each algorithm. In each run, the algorithm initially makes rapid improvement. After this, improvement begins to take more time, and eventually the algorithm fixates or niches on a local optimum.

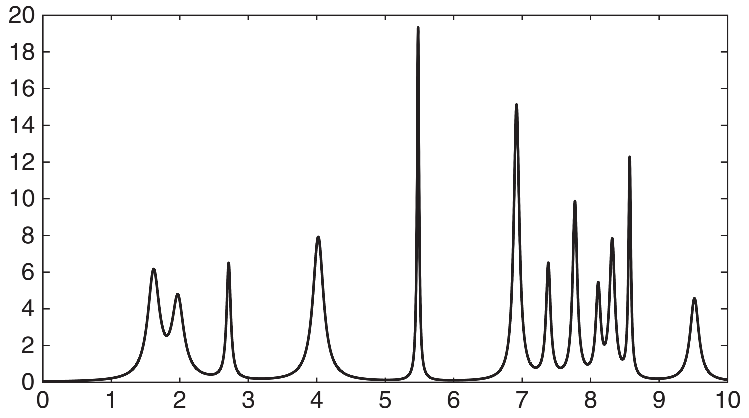


Fig. 4.6. The Shekel function specified by Table 4.1.

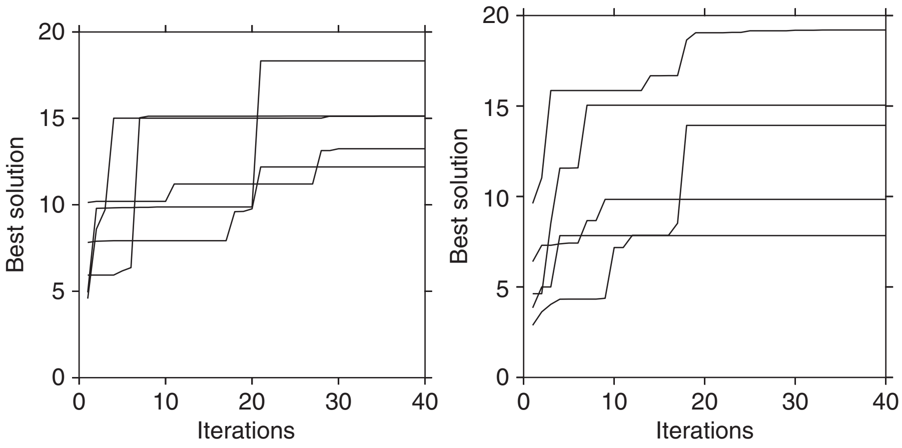


Fig. 4.7. Progress toward the Shekel problem solution using GAs.

Using Real Numbers Instead of Bit Strings

If the function $f(x)$ being maximized is defined on several variables, that is, x is a vector, $x = (x_1, x_2, \ldots, x_d)$, then the natural bit string to take as the chromosome is just the concatenation of the bit representations of each component.

However, it may be preferable to treat the components of x directly as real numbers. For example, the d-dimensional vector itself can serve as the chromosome. Crossover is then an operation that exchanges components. But if the components x_i are real numbers, what can mutation be?

Actually, we have already seen the canonical mutation operation for real numbers in Chapter 3 when we were sampling continuous distributions. Namely, select the perturbation of x_i uniformly at random from a small interval $(x_i - \delta, x_i + \delta)$ around x_i.

Continuing in this vein, it is also possible to define a crossover operation between two real numbers and thereby avoid bit strings altogether. A reason to do so is to avoid programming bit operations, which may be difficult to do in some computer languages (such as Matlab) and to avoid translation between bit strings and the real numbers they represent. A simple operation between positive floating-point numbers x and y that is similar to one-point crossover is this:

$$\begin{aligned}&\text{select gainsize } g \sim U(0,1)\\ &\text{put } x' = (1-g)x + gy \text{ and } y' = (1-g)y + gx.\end{aligned}$$

It is easy to see that if $y < x$, then $y < x' < x$ and $y < y' < x$.

An operation even closer to one-point crossover is the following:

$$\begin{aligned}&\text{select grainsize } g \sim U(0,1)\\ &\text{put } t_x = g{*}\text{frac}(x/g) \text{ and } t_y = g{*}\text{frac}(y/g)\\ &\text{put } x' = (x - t_x) + t_y \text{ and } y' = (y - t_y) + t_x.\end{aligned}$$

In this operation, the t_x are the "tailbits" of x defined by g, and the t_y are the same for y. Then $x - t_x$ are the "leading bits" of x, and $y - t_y$ are the same for y.

4.5 An Application of GA to the Permanent Problem

Given a square matrix $C = (c_{ij})$ of size n, its determinant is defined as

$$\det C = \sum_{\sigma} \text{sign}(\sigma) \prod_{i=1}^{n} c_{i,\sigma(i)}.$$

In this formula, σ is a permutation of the first n integers, $\{1, 2, \ldots, n\}$, and the sum extends over all such permutations. Furthermore, $\text{sign}(\sigma)$ is defined to be $+1$ if σ is an even permutation and -1 if σ is an odd permutation (a permutation is even/odd if it can be expressed as an even/odd number of transpositions, or two-point exchanges). One sees that the product part of the formula proceeds down the rows of the matrix, $i = 1, 2, \ldots, n$, choosing one term in each row from whatever column is dictated by the permutation $\sigma(i)$ so that each column is represented once and only once. This permutation definition leads to the familiar expansion-by-minors formula for determinants. And thanks to the sign(σ) factor, it also leads to a Gaussian elimination expansion technique. In this form, the computational complexity can be shown to be O(n^3) (that is, the number of computations is bounded by Kn^3 for some constant K).

Related to the determinant of a matrix is the *permanent* of a matrix. It is defined in the same way, except that the sign(σ) factor is omitted:

$$\operatorname{perm}(C) = \sum_{\sigma} \prod_{i=1}^{n} c_{i,\sigma(i)}.$$

For example, since the permutations of $\{1, 2, 3\}$ are 123, 132, 213, 231, 312, and 321, we have

$$\operatorname{perm}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = c_{11}c_{22}c_{33} + c_{11}c_{23}c_{32} + c_{12}c_{21}c_{33} \\ + c_{12}c_{23}c_{31} + c_{13}c_{21}c_{32} + c_{13}c_{22}c_{31} = 3!.$$

The permanent is mainly used in combinatorics.[1]

Unfortunately, there is no short-cut method known for computing the permanent, and its computational complexity is O($n!$).

Now consider the space of $n \times n$ matrices whose terms are only 0 or 1. These are much easier to compute than unrestricted matrices, because as soon as a 0 is encountered in a product corresponding to some permutation, 0 may immediately be written for that product and calculation may proceed to the next permutation. For example, one immediately sees that

$$\operatorname{perm}\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = 0$$

because the last row has all 0's.

[1] For example, it gives the number of perfect matchings in a bipartite graph. This is a graph for which the vertices can be partitioned into two sets A and B such that every edge has one incidence in A and one in B. Figure 3.8 is bipartite. A perfect matching is a matching that covers all vertices of the graph. That is, every vertex of the graph is incident to exactly one edge of the matching. If E is the edge matrix of the graph, with e_{ij} equal to 1 if there is an edge joining vertex i to vertex j and 0 otherwise, then perm(E) is equal to the number of perfect matchings in the graph.

```
Matlab
> % mat is an nxn 0/1 matrix
> % used is an n-dim. array
> function y=permrecurse(level,n,used,mat,v)
> if level == n+1
     y=v+1;
  else
     for i=1:n
        if used(i) == 0 & mat(level,i) ~= 0
        used(i)=1;
        v = permrecurse(level+1,n,used,mat,v);
        used(i)=0;
        end
     end
     y=v;
  end
```

```
Typical call to permrecurse
> n=3;
> used=zeros(1,n);
> a=[[1,0,1];[0,1,1];[1,1,1]]
> permrecurse(1,n,used,a,0)
```

Fig. 4.8. Recursive Matlab algorithm for computing a 0/1 permanent.

In Figure 4.8 we give a remarkably short, recursive program for calculating the permanent of a 0/1 matrix. In Figure 4.9 we give a nonrecursive algorithm for calculating the permanent as an example for languages in which recursion is not possible.

The optimization problem is this: among the set of $n \times n$ matrices with 0/1 entries and having exactly m 1's, what is the largest possible permanent and what matrix gives the maximum value? We shall refer to this as the $n : m$ *permanent problem*. For example, for the $3 : 5$ permanent problem, the maximum permanent is 2 and it is given by

$$\text{perm}\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 2.$$

Several rearrangements of this matrix also give 2; thus

$$\text{perm}\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix} = \text{perm}\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = 2.$$

Like the TSP, the $m : n$ permanent problem is also well-suited for testing optimization algorithms. In this case, not only does the solution space grow exponentially, but also the “fitness landscape” defined by the permanent becomes more deceptive. That is, states with high permanent value are surrounded by states with much lower permanent value. The “fitness landscape” is characterized by extremely small peaks surrounded by huge flat plains.

Permanent Calculator for 0/1 Matrices (f77)

```
c23456789
      PROGRAM PERMNR
C
      INTEGER N,A(15,15)
      INTEGER PER, PERM(15), OD(15)
      COMMON A,PERM,OD,PER
C
      N=3
      DATA A(1,1),A(1,2),A(1,3)/1,0,1/
      DATA A(2,1),A(2,2),A(2,3)/0,1,1/
      DATA A(3,1),A(3,2),A(3,3)/1,1,1/
C
      DO 10, I=1,N
   10 PERM(I)=I
      CALL ODOMETER(N)
C the permanent of A is now in PER
      print *,'PER= ',PER
      STOP
      END
C
      SUBROUTINE CP(N)
      INTEGER N,A(15,15)
      INTEGER PER, PERM(15), OD(15)
      COMMON A,PERM,OD,PER
      DO 10, I=1,N
         IF (A(I,PERM(I)).EQ.0) RETURN
   10 CONTINUE
      PER = PER+1
      RETURN
      END
C
      SUBROUTINE ODOMETER(N)
      INTEGER N,A(15,15)
      INTEGER PER, PERM(15), OD(15)
      COMMON A,PERM,OD,PER
      DO 10, I=1,N
   10 OD(I) = 0
      I = N-1
   20 CONTINUE
        IF (OD(I) .GT. N-I) THEN
          OD(I) = 0
          I=I-1
          IF (I .LT. 1) GOTO 100
        ELSE
          K=PERM(I)
          PERM(I)=PERM(I+1)
          PERM(I+1)=K
          IF (OD(I) .EQ. N-I) THEN
            DO 30, J=I+1,N-I
              K=PERM(J)
              PERM(J) = PERM(J+1)
   30       PERM(J+1) = K
          END IF
          CALL CP(N)
          I=N-1
        END IF
        OD(I)=OD(I)+1
        GOTO 20
  100 RETURN
      END
```

Fig. 4.9. Fortran 77 program for evaluating a permanent.

GA Solution

To construct a genetic algorithm, we must (1) define the chromosomes, (2) define the fitness function, (3) and define the genetic operators: mutation and crossover. As chromosomes we take $n \times n$ 0/1 matrices having exactly m 1's. The population will consist of 12 such chromosomes. Let the permanent of a matrix be its fitness. By maximizing fitness we will be maximizing the permanent.

For mutation we take "north, south, east, west (NSEW) neighbor swap with wraparound." This means that in the matrix C to undergo mutation:

1. select an entry uniformly at random in C, say c_{ij},
2. choose a direction N, S, E, W uniformly at random and interchange the element in that direction with c_{ij}.

For example, if north is chosen, then interchange $c_{i-1,j}$ with c_{ij}. Wraparound means that if $i = 1$, then take as the north neighbor the element $c_{n,j}$. Or if $j = n$, take as the east neighbor c_{i1}, and so on. If the chosen neighbor and c_{ij} are the same, both 0's or both 1's, then make another choice of direction. If all neighbors are the same, skip the mutation operation. Figure 4.10 illustrates the neighbors (labeled with circles in the figure) of the darkened squares. The one on the upper right corner of the matrix has its "north" neighbor on the bottom row and its "east" neighbor in the first column.

Note that this operation preserves the number of 1's in the matrix C.

For the crossover operation we take "templated global swap." Let the matrices A and B be selected for mating. String out their matrix elements, row by row, into two long 1-dimensional arrays. Do this with wraparound meaning that the end of each string is joined to the beginning, so that the next element after a_{nn} is a_{11} and similarly for B. Now choose the same starting point in each string at random. Move term by term along each string until the first time the elements differ; string A is a 0, say, and string B is a 1. Swap these elements.

Continuing from there, move along the strings until you first encounter a difference in the elements the other way around: string A is a 1 and string B is a 0. Again swap. In effect, two terms of A (B) have been globally interchanged based on the template of B (A).

As before, the count of 1's in the matrices mated has been preserved.

These specific operations may now be programmed into either of the GA prototype algorithms. The results of six runs of GA Algorithm

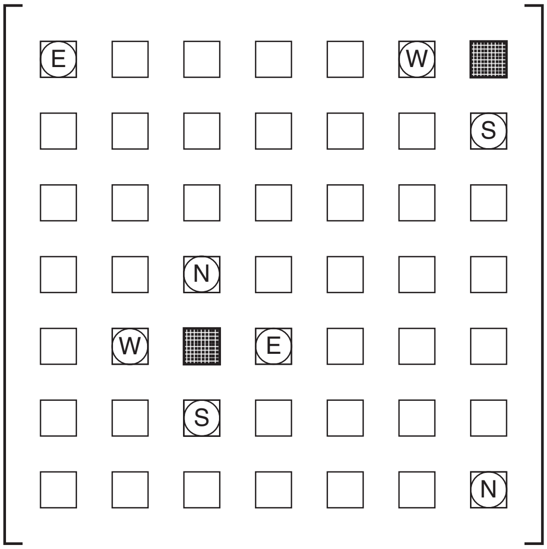


Fig. 4.10. Illustration of mutation neighborhood for permanent problem.

2 working on the 14:40 permanent problem are shown in Figure 4.11. It is known that the maximum possible permanent for this problem is 2592.

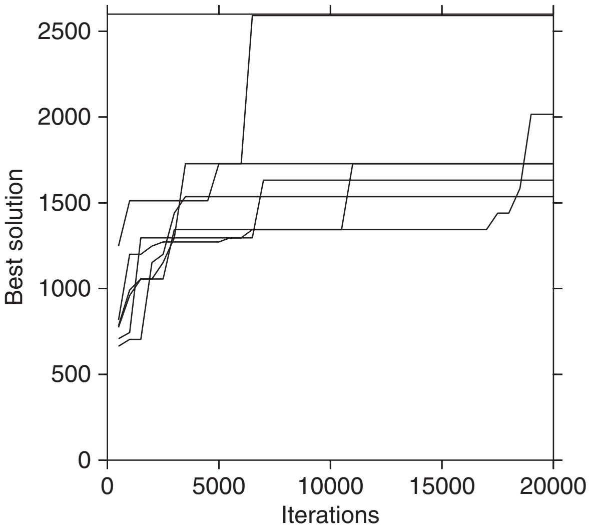


Fig. 4.11. Solution history of GA 2 on the 14:40 permanent problem.

Notes

Simulated annealing and genetic algorithms are just two examples of Monte Carlo optimization methods. Many other methods exist under the general name of evolutionary algorithms (see [Ash06, BFM97]).

Most optimization algorithms are plagued with the problem of niching. Adding a random component to a deterministic optimization algorithm is one way of dealing with niching, for example some form of restarting. Monte Carlo optimization methods have a built-in method for dealing with niching. However, this sometimes comes at a cost, since the expected time to find the optimum can be infinite (see Problem 1).

A good source of information on the TSP is the TSPBIB, currently located at the URL

`http : //www.densis.fee.unicamp.br/ moscato/TSPBIB_home.html`.

For many TSP problem instances see the TSPLIB resource at the URL

`http : //www.iwr.uni − heidelberg.de/groups/comopt/software/`
`TSPLIB95/`.

Both simulated annealing and genetic algorithms can be modified to handle constrained optimization problems in a natural way. The time-varying aspect of SA makes a dynamic penalty method a natural choice. A very nice analysis of an algorithm called “Compressed Annealing” can be found in [OBH04].

Problems: Chapter 4

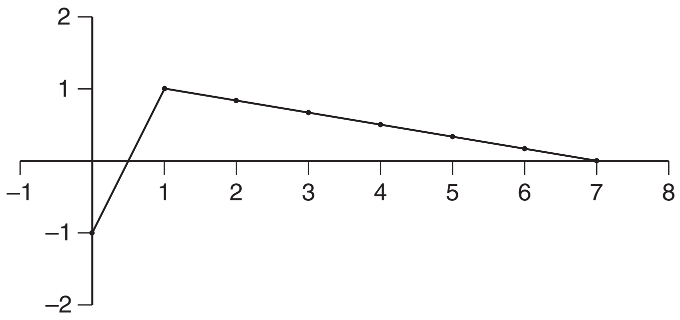


Fig. 4.12. Three-bit Sandia Mountain problem (defined only on the integers).

1. (5) (a) The n-bit *Sandia Mountain problem* is to find the minimum value of the function f defined as follows on $\{0, 1, 2, \ldots, N\}$:

$$f(x) = \begin{cases} 2x - 1, & \text{if } x = 0 \text{ or } 1, \\ \frac{N-x}{N-1}, & \text{if } x = 1, 2, \ldots, N, \end{cases}$$

where $N = 2^n - 1$; see Figure 4.12. Consider the 2-bit Sandia Mountain problem (so the state space is $\{0, 1, 2, 3\}$, and $f(0) = -1$, $f(1) = 1$, $f(2) = 1/2$, and $f(3) = 0$). Verify that using SA with minimal neighborhoods and inverse log cooling, $T = 1/\log(t + 1)$, the transition probability matrix is

$$\begin{bmatrix} 1 - \frac{1}{2}e^{-2/T} & \frac{1}{2}e^{-2/T} & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & \frac{1}{2}e^{-1/2T} & \frac{1}{2}(1 - e^{-1/2T}) & \frac{1}{2} \\ 0 & 0 & \frac{1}{2}e^{-1/2T} & 1 - \frac{1}{2}e^{-1/2T} \end{bmatrix}.$$

(b) Write an annealer for this problem. Starting from $x = 3$, run it to find the globally minimum value, -1 at $x = 0$; histogram the number of iterations until the minimum is found. If the solution is not found in 1000 trials, stop and add 1000 to the histogram. Now repeat this, except allow 10,000 trials before stopping.
(c) What is the expected number of iterations to find the solution starting from $x = 3$?
Note: this should be like the St. Petersburg problem.

2. (3) Write an annealer to solve the 4-bit Sandia Mountain problem (see Problem 1 above for a definition). Try various neighborhood structures including an all-connected (maximal) one. Which one seems to work best? Why?

3. (4) Consider the n-bit Sandia Mountain problem where n is quite large (so that 2^n, the number of states is very large). Explore the benefits and drawbacks of a minimal neighborhood system versus an all-connected (maximal) neighborhood system.

item	1	2	3	4	5	6	7	8	9
value	189	149	177	158	140	192	155	165	160
cost	920	1021	1065	1038	1041	1089	1016	1081	920

item	10	11	12	13	14	15	16	17	18
value	102	134	100	174	188	102	166	135	101
cost	1035	977	1039	976	979	926	1085	931	917

Table 4.2. Table of values and costs for the knapsack problem.

4. (6) (Knapsack problem) Let $x = (x_1, x_2, \ldots, x_n)$ be a 0/1 vector ($x_i = 0$ or 1 for all i). Given the data in Table 4.2, use a genetic algorithm to maximize

$$\sum_{i=1}^{n} \text{value}_i \cdot x_i,$$

provided that

$$\sum_{i=1}^{n} \text{cost}_i \cdot x_i \le 10{,}000.$$

Notice that to do this, it is necessary somehow to ensure that the GA operations preserve the constraint.

5. (5) Use a genetic algorithm to solve the same knapsack problem as above, but this time allow solutions to violate the constraint, and apply a penalty to the *infeasible* solutions. Explore various penalty methods, including a dynamic penalty that increases the penalty over the course of the run of the algorithm.

6. (6) Anneal four particles on the line confined to the segment between -1 and 1. The state of the system is the vector of their positions, $\mathbf{x} = [\, x_1 \quad x_2 \quad x_3 \quad x_4 \,]$. Take energy to be

$$E(\mathbf{x}) = \sum_{i<j} \Phi(r_{ij}) \qquad r_{ij} = |x_i - x_j|$$

and $\Phi(r) = 4000 \left[(0.1/r)^{12} - (0.1/r)^{6}\right]$. For a perturbation, choose one of the particles equally likely and move it according to a Gaussian with mean 0 and standard deviation of 0.2 with reflection at -1 and $+1$. Take temperature to be $T = 0.04$ in one run and 0.08 in another. Histogram energies.

7. (5) Write a simulated anneal solver for the permanent 8:20 problem. (The temperature range should include $T = 2$. Note: the size of the solution space is $\binom{8^2}{20} = 19{,}619{,}725{,}782{,}651{,}120$, while the number of these points that are optimal solutions is $2(8!/(3!)^2)^2 = 2{,}508{,}800$, this is about 1 part in 10^{10}.)

8. (6) Write a genetic algorithm for solving the permanent 8:20 problem. Be sure to indicate (a) what a chromosome is, (b) what fitness is, (c) what the genetic operations are.

9. (5) Write an annealer for the TSP on the 16 cities defined below. (Note: the solution space has $(16-1)!/2 = 653{,}837{,}184{,}000$ tours.)

city1 = (12,12), city2 = (18,23), city3 = (24,21),

city4 = (29,25), city5 = (31,52), city6 = (36,43),
city7 = (37,14), city8 = (42,8), city9 = (51,47),
city10 = (62,53), city11 = (63,19), city12 = (69,39),
city13 = (81,7), city14 = (82,18), city15 = (83,40),
city16 = (88,30).

10. (7) Write a genetic algorithm for the 16 city TSP above. Be sure to indicate (a) what a chromosome is, (b) what fitness is, (c) what the genetic operations are. Note: there is no "natural" crossover operation for TSP tours that the authors know of. Be creative and if you find a good one, write us!

a	b	c	a	b	c
11.380	7.928	0.1515	4.714	7.480	0.1339
7.692	3.739	0.0900	13.688	1.535	0.0715
11.687	2.389	0.2139	9.606	8.236	0.1788
12.746	3.003	0.1783	9.565	5.249	0.1780
9.786	2.132	0.0551	3.023	3.773	0.1395
13.985	0.620	0.1181	13.905	3.929	0.2164

Table 4.3. Parameters for a 12-term Shekel function.

11. (5) Write an annealer to find the maximum of the Shekel function whose 12-term parameters are given in Table 4.3.

12. (5) Write a genetic algorithm to find the maximum of the Shekel function whose 12-term parameters are given in Table 4.3.

13. (6) The following is a table of the US population. Find the parameters A, r, and K of the equation

$$y = \frac{Ae^{rt}}{1 + \frac{A}{K}e^{rt}}$$

that best fits the population data. Use least squares between the curve and the population data as the measure of closeness.

U.S. Population Census*					
1790	3929214	1860	31433321	1930	122775046
1800	5308483	1870	39818449	1940	131669275
1810	7239881	1880	50155783	1950	151325798
1820	9638453	1890	62947714	1960	179323175
1830	12866020	1900	75994575	1970	203302031
1840	17069453	1910	91972266	1980	226545805
1850	23191876	1920	105710620	1990	248709873

* Source: Time Almanac Reference Edition, 1994.

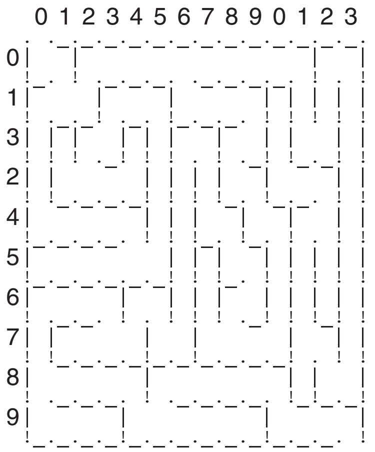
14. (6) (TSP with time windows – TSPTW) The TSP with time windows is a variant of the TSP in which the salesman must visit a specific city during a given window of time. Instances of the TPSTW can be found at

    `http : //myweb.uiowa.edu/bthoa/TSPTWBewnchmarkDataSets.htm`.

    Write an annealer to solve the TSPWT. Notice that the TSPTW is a constrained optimization problem, and thus your algorithm will need some way of handling constraints, either by ensuring that the constraints are always satisfied or by applying some penalty to the infeasible states.

15. (6) Figure 4.13 shows a 10×14 maze along with its definition. Write an annealer to solve it (or some other maze of your choosing). The tasks are encoding a solution, inventing a perturbation of a solution, and defining an "energy" for a solution that improves as solutions get closer to the end.

16. (7) Write a genetic algorithm solver for the maze above. Indicate (a) what a chromosome is, (b) what fitness is, (c) what the genetic operations are.



```
rows= 10, columns= 14

e 0, s  1    e 1, s 0    e 0, s 1     e 0,  s 0     e 0, s  0
e 1, s  0    e 1, s 0    e 0, s 1     e 0,  s 1     e 0, s  1
e 0, s  0    e 0, s 0    e 0, s 1     e 0,  s 1     e 0, s  1
e 0, s  1    e 1, s 1    e 0, s 1     e 1,  s 0     e 0, s  1
e 0, s  1    e 1, s 0    e 1, s 0     e 0,  s 0     e 1, s  0
e 0, s  1    e 1, s 0    e 1, s 0     e 1,  s 0     e 0, s  0
e 0, s  0    e 0, s 0    e 1, s 0     e 1,  s 0     e 0, s  1
e 0, s  1    e 1, s 0    e 0, s 1     e 1,  s 0     e 0, s  1
e 0, s  1    e 0, s 0    e 1, s 0     e 0,  s 0     e 0, s  1
e 0, s  1    e 1, s 1    e 0, s 1     e 1,  s 1     e 0, s  1
e 0, s  1    e 1, s 0    e 1, s 0     e 1,  s 0     e 1, s  0
e 1, s  0    e 0, s 1    e 0, s 0     e 1,  s 0     e 1, s  1
e 0, s  0    e 1, s 1    e 1, s 0     e 1,  s 1     e 0, s  1
e 1, s  0    e 1, s 0    e 1, s 0     e 1,  s 0     e 1, s  1

e 0, s  0    e 1, s 0    e 0, s 1     e 1,  s 0     e 0, s  1
e 0, s  1    e 0, s 1    e 0, s 1     e 0,  s 1     e 0, s  1
e 1, s  1    e 0, s 1    e 0, s 1     e 0,  s 1     e 0, s  1
e 0, s  0    e 0, s 1    e 0, s 1     e 0,  s 1     e 1, s  1
e 0, s  1    e 1, s 1    e 0, s 1     e 1,  s 1     e 0, s  1
e 1, s  0    e 1, s 0    e 1, s 1     e 0,  s 1     e 0, s  1
e 0, s  1    e 1, s 0    e 1, s 0     e 1,  s 1     e 0, s  1
e 0, s  1    e 1, s 0    e 1, s 0     e 0,  s 1     e 0, s  1
e 0, s  1    e 0, s 1    e 0, s 1     e 0,  s 1     e 0, s  1
e 1, s  0    e 1, s 0    e 1, s 0     e 0,  s 1     e 1, s  1
e 1, s  0    e 0, s 1    e 1, s 0     e 1,  s 1     e 0, s  1
e 1, s  0    e 0, s 1    e 1, s 0     e 0,  s 0     e 0, s  1
e 1, s  0    e 1, s 0    e 1, s 0     e 1,  s 0     e 0, s  1
e 1, s  0    e 1, s 0    e 1, s 0     e 1,  s 0     e 1, s  1
```

Fig. 4.13. A 10×14 maze, start at $(0,0)$, end at $(9,13)$. Below the maze is its computer definition. There are 140 "cells" to the maze, and each cell has an east (e) wall, either open (0) or closed (1), and a south (s) wall. Read down the columns left to right for the cells going across the maze top to bottom.

5 Random Walks

While a Markov chain can be considered a random walk (on an appropriate state space), a random walk is not always an instance of a Markov chain. For example, a random walk's next step could depend on the entire history of the walk up to that time. This is the case for self-avoiding walks, which have applications in the study of macromolecules.

Random walks arise in the motion of particles under collision (such as *Brownian motion*), in gambling problems (the fortune of a (perhaps unfortunate) gambler), and in mathematical models in finance (such as the pricing of options).

5.1 Diffusion

In 1828, the botanist Robert Brown published a pamphlet entitled, "A Brief Account of Microscopical Observations on the Particles Contained in the Pollen of Plants." He had observed, through a microscope, pollen grains suspended in water and was amazed to see that they underwent a constant and erratic motion. We now know that the grains move under the unseen influence of their collisions with water molecules. The water molecules themselves are under constant erratic motion due to collisions with each other. Their average speed, and hence their kinetic energy, increases with temperature. Energy is propagated through the medium via these collisions.

The water molecules also move from place to place within the medium as a result of the collisions. So do the pollen grains, but much more slowly. This type of transport is called *diffusion*. The motion can be described, and modeled, as a *random walk*.

To keep things simple, consider the random walk on the lattice of integers on the real line by a single particle. The particle starts at the origin and takes one step to the right with probability p and one step to the left with probability $q = 1 - p$. If $p = q = 1/2$, the choice could be made by the flip of a fair coin. In Figure 5.1 we show the histogram of the endpoints of 200,000 random walks each of 30 steps. The figure uses the following code:

```
 Matlab
> nSteps = 30;
> nTrials = 200000;
> H=zeros(1,nTrials);
 for j=1:nTrials
  % do all nSteps coin tosses
  T = 2*(rand(1,nSteps) < 0.5 ) -1;
  H(j) = sum(T);
> end
> hist(H)
```

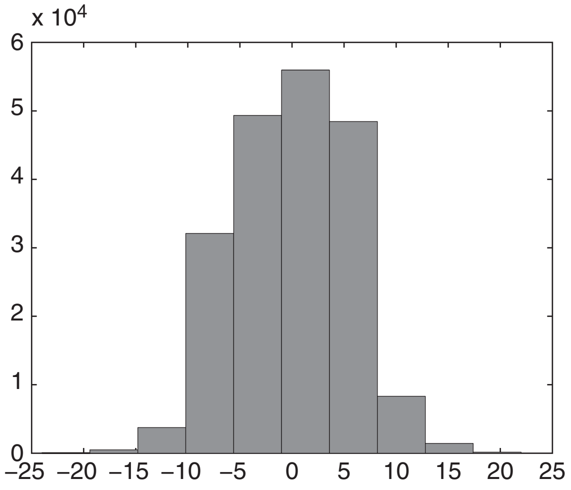


Fig. 5.1. Endpoints of 200,000 walks of 30 steps.

As we should expect by the central limit theorem, the histogram approximates the normal distribution. By increasing the number of steps and making them smaller we can derive the equations of 1-dimensional diffusion.

Diffusion as the limit of a Random Walk

For simplicity, assume that time is divided into discrete periods Δt and in each such period a particle moves randomly along a line, one step Δx to the right, with probability p, or one step Δx to the left, with probability $q = 1 - p$. After n time periods the particle lies somewhere in the interval from $-n(\Delta x)$ to $n(\Delta x)$ relative to its starting point, taken as the origin 0.

Let $p(m, n)$ denote the probability that the particle has been displaced to position $x = m(\Delta x)$, m steps right of the origin, after n time periods, $t = n(\Delta t)$. We wish to calculate $p(m, n)$. It will help to recognize that our random walk with n steps is something like tossing n weighted coins. For every coin that lands heads we step right, and for tails we step left. Let r be the number of steps taken to the right and l the number left. Then to be at position $m(\Delta x)$ it must be that their difference is m,

$$m = r - l, \qquad \text{where} \qquad n = r + l.$$

Thus r can be given in terms of m and n by adding these two equations, and l is given by subtracting,

$$r = \frac{1}{2}(n + m) \qquad \text{and} \qquad l = \frac{1}{2}(n - m).$$

As in a coin-tossing experiment, the number of ways of selecting r moves to the right out of n possibilities is given by

$$\binom{n}{r} = \frac{n!}{r!\,(n-r)!}.$$

If the probabilities of going left and of going right are q and p as above, then

$$p(m, n) = \text{probability of } r \text{ steps right} = \binom{n}{r} p^r q^{n-r}, \qquad r = \frac{1}{2}(n+m).$$

This is the *binomial* distribution (as discussed in Section 2.3).

If the random walk experiment with $n = 100$ steps were conducted a large number of times, then a histogram of the resulting particle positions would closely approximate a normal distribution (by the central limit theorem). Alternatively, the same result pertains to the fate of a large number of particles randomly walking at the *same* time, each taking 100 steps, provided they may slide right past each other without collisions.

We next determine which normal distribution is the limit, that is, what are the mean and variance of the limiting normal distribution. The expected displacement is given by

$$E(m) = E(2r - n) = 2E(r) - n = 2np - n = (2p - 1)n.$$

This is the *drift* of the random walk. The variance of the displacement is

$$\text{var}(m) = \text{var}(2r - n) = 4\text{var}(r) = 4npq.$$

Hence the *root mean square (rms)* displacement, which quantifies the dispersion, is

$$\sqrt{\text{var}(m)} = \sqrt{4npq}.$$

Notice that this is just the standard deviation of the displacement. In the case where $p = q = 1/2$ we see that

$$\text{rms displacement} = \sqrt{n}.$$

Hence particles disperse in proportion to the square root of time. To traverse a distance twice as far requires four times as much time.

To continue with the normal approximation, we now fix $p = q = 1/2$. Recall that the probability that a normally distributed observation will fall within an interval of width dm centered at m is, approximately,

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(m-\mu)^2}{2\sigma^2}}\, dm,$$

where μ is the mean and σ is the standard deviation of the distribution. On the other hand, $p(m, n)$ is the probability that the walk will end between $m - 1$ and $m + 1$, an interval of width 2, and from above, its mean is 0 and standard deviation is $\sqrt{n}$. Hence

$$p(m, n) \approx \frac{1}{\sqrt{2\pi n}} e^{-\frac{m^2}{2n}}(2) \approx \sqrt{\frac{2}{\pi n}} e^{-\frac{m^2}{2n}}. \tag{5.1}$$

Note that here we are using the continuous normal distribution to approximate the discrete random walk distribution. This is the reason we set $dm = 2$, to concentrate the spread-out area of the normal distribution over an interval of 2 into a single central point.

Jump to the Continuum

Our last refinement is to let Δx and Δt tend to 0 to obtain a continuous version of $p(m,n)$. Of course, without care, $p(m,n)$ will approach zero as well, because the probability will spread out over more and more values of m. But since each value of m corresponds to a probability over a width of $2\Delta x$, we take the quotient of $p(m,n)$ by this width. Here we are doing the opposite of what we did above; we are spreading out the discrete probability at the central point of an interval over the entire interval, so we must divide. Let $u(x,t)$ denote the probability that the particle lies in an interval of width $2(\Delta x)$ centered at x at time t. Then

$$u(x,t) = \frac{p(\frac{x}{\Delta x}, \frac{t}{\Delta t})}{2(\Delta x)} = \frac{1}{2(\Delta x)}\sqrt{\frac{2}{\pi(\frac{t}{\Delta t})}}e^{-\frac{(x/\Delta x)^2}{2(t/\Delta t)}}.$$

And upon simplification,

$$u(x,t) = \frac{e^{-(x^2/4(\frac{\Delta x^2}{2\Delta t})t)}}{\sqrt{4\pi\left(\frac{\Delta x^2}{2(\Delta t)}\right)t}}.$$

Now keeping the ratio

$$D = \frac{\Delta x^2}{2(\Delta t)}$$

fixed as Δx and Δt tend to 0, we obtain the Gaussian distribution

$$u(x,t) = \frac{e^{-\frac{x^2}{4Dt}}}{\sqrt{4\pi Dt}}.$$

The parameter D is called the *diffusion coefficient* or *diffusivity* and has units of area divided by time. Diffusivity depends on the solute, the solvent, and the temperature among other things. As to why the ratio $\Delta x^2/\Delta t$ should be kept fixed (and not $\Delta x/\Delta t$ for example), we have noted above that under the process of diffusion, displacement is proportional to the square root of time: in half the time, the process can be expected to go only one-fourth the distance. In this way we avoid infinite velocities as $\Delta t \to 0$.

This function $u(x,t)$ satisfies the diffusion equation

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}. \tag{5.2}$$

Solutions to this partial differential equation are said to be *harmonic functions*.

Recurrence

We have already seen that a walk disperses at the rate of $\sqrt{n}$. It might be supposed that particles wander away, never to return. But for the 1-dimensional and 2-dimensional random walks, this is not so. In fact, it can be shown that given any lattice point, with probability one the walk will attain that point eventually as $n \to \infty$. This means in particular that the walk will reach any predetermined value M, for example $M = 100$ or $M = 1{,}000{,}000$ or $M = -9{,}192{,}129$ and so on. It also means that the walk will return to any point and do so infinitely often, for example the origin. This property is known as *recurrence*, and a walk that satisfies this property is said to be *recurrent*. In Figure 5.2 we show why this is so for a 1-dimensional lattice walk.

Let R be the probability that the walk starting from the origin will return. We decompose on the first step; to the left or right. But we may assume that it is to the right by symmetry: if it were to the left, we could just rotate the line 180° and things would be the same. Now from $x = 1$, with probability 1/2 the next step is to the left and we are back at the origin. Otherwise, with probability 1/2 the next step is to $x = 2$. But now, to get back to the origin, we must get back to $x = 1$. Since the line is infinite, the probability of that is exactly R. So we have the following:

$$R = \frac{1}{2} + \frac{R}{2}\left[\text{times the prob. of reaching } x = 0 \text{ from } x = 1\right]. \tag{5.3}$$

But the probability of reaching $x = 0$ from $x = 1$ is exactly the right-hand side of this equation; therefore

$$\begin{aligned} R &= \frac{1}{2} + \frac{R}{2}\left[\frac{1}{2} + \frac{R}{2}\left[\frac{1}{2} + \frac{R}{2}[\cdots]\right]\right] \\ &= \frac{1}{2} + \frac{1}{2}\left(\frac{R}{2}\right) + \frac{1}{2}\left(\frac{R}{2}\right)^2 + \frac{1}{2}\left(\frac{R}{2}\right)^3 + \cdots \\ &= \frac{1}{2}\left[\frac{1}{1 - R/2}\right]. \end{aligned}$$

So $2R(1 - R/2) = 1$, from which it follows that $R = 1$.

Fig. 5.2. Return to the origin is certain for a 1-dimensional walk.

5.2 Brownian Motion

The 1-dimensional random walk on the lattice of integers studied in the previous section is only a very crude approximation to Brownian motion. It is somewhat remarkable that by allowing the step sizes to become small and the steps to occur more frequently, we are able to derive the law of diffusion, if not Brownian motion itself. But what exactly is Brownian motion?

The mathematical theory of Brownian motion was put on firm ground by Norbert Wiener. Consequently, Brownian motion is often called a *Wiener process*. It is defined by three axioms; letting W_t be the random variable of the location of the particle at time t (here t is a continuous variable), then:

- $W_0 = 0$,
- for $t_1 < t_2 < \cdots < t_n$, the differences
$$W_{t_2} - W_{t_1}, \ldots, W_{t_n} - W_{t_{n-1}}$$
are independent, and
- for $s < t$, $W_t - W_s$ is normally distributed with mean 0 and variance $c(t - s)$, where $c > 0$ is a fixed constant.

For a start, we see that Brownian motion step sizes are normally distributed with increasing variance over longer times. By the independence assumption, we see that to obtain Brownian motion in higher dimension, for example two dimensions, we can simply make the steps of the walk fall alternately in the x-direction and then in the y-direction. We could also chose a direction at random and take a normally distributed step size in that direction. The upper left graph in Figure 5.3 is a simulated 2-dimensional Brownian motion obtained by this second method. At every step, the new direction is randomly chosen and the step size is chosen according to a normal distribution with fixed standard deviation. This corresponds to observing the position of the particle at fixed time intervals.

The graph in the lower right of the figure shows the endpoints of 400 of these walks. It is a visualization of the probability density of the distance the particle achieves after 2000 steps. As noted above, this is a 2-dimensional normal density distributed symmetrically around the origin.

```
Matlab
> sig=2; % step size std dev
> nSteps = 2000; % # steps in each walk
> nSteps = 2*fix(nSteps/2); % assure even
> % box-muller for N(0,1) random variables
> u1=rand(nSteps/2,1);
    % U[0,1) vector nSteps/2 long
> u2=rand(nSteps/2,1);
> x1 = cos(2*pi*u1).*sqrt(-2*log(u2));
  %N(0,1) vector
> x2 = sin(2*pi*u1).*sqrt(-2*log(u2));
> % concatenate x1 and x2 into h
> h(1:nSteps/2) = x1;
> h(1+nSteps/2:nSteps) = x2;
> x(1)=0; y(1)=0; %start at the origin
> for i=2:nSteps
    theta = 2*pi*rand; % random direction
    x(i)=x(i-1)+sig*h(i)*cos(theta);
    y(i)=y(i-1)+sig*h(i)*sin(theta);
  end
> %typical walk
> figure(1)
> subplot(2,2,1);
> plot(x,y);
> title('Typical walk for nSteps steps');
> xlabel('X position');
> ylabel('Y position');
> % do the walk from subStart to subEnd
> sub1Start = 500;
> sub1End = 1500;
> k = 1;
  for i=sub1Start:sub1End
    xsub1(k) = x(i);
    ysub1(k) = y(i);
    k = k+1;
  end
> figure(1)
> subplot(2,2,2);
> plot(xsub1,ysub1);
> sub2Start = 1000;
> sub2End = 1300;
```

```
continued...
> k = 1;
> for i=sub2Start:sub2End
    xsub2(k) = x(i);
    ysub2(k) = y(i);
    k = k+1;
  end
> figure(1)
> subplot(2,2,3);
> plot(xsub2,ysub2);
> nWalks = 400; % # walks for density
> for j=1:nWalks
    u1=rand(nSteps/2,1);
    u2=rand(nSteps/2,1);
    x1 = cos(2*pi*u1).*sqrt(-2*log(u2));
    x2 = sin(2*pi*u1).*sqrt(-2*log(u2));
    h(1:nSteps/2) = x1;
    h(1+nSteps/2:nSteps) = x2;
    x(1)=0; y(1)=0;
    for i=2:nSteps
      theta = 2*pi*rand;
      x(i)=x(i-1)+sig*h(i)*cos(theta);
      y(i)=y(i-1)+sig*h(i)*sin(theta);
    end
    Endx(j) = x(nSteps);
    Endy(j) = y(nSteps);
  end
> %density plot
> figure(1);
> subplot(2,2,4);
> scatter(Endx,Endy,3);
> title('Density Plot for nSteps steps');
> xlabel('X end');
> ylabel('Y end');
```

Self-Scaling of Brownian Motion

Notice that the path of the walk has the pattern of separated clusters. Most of the time the step size is approximately equal to the standard deviation. This forms the cluster. Occasionally there is a much larger step, and sometimes more than one within a few steps of each other. These form the tendrils of the walk that connect the clusters.

This pattern is repeated at all scales. The graphs in the upper right and lower left are successive zooms on a portion of the walk. They show

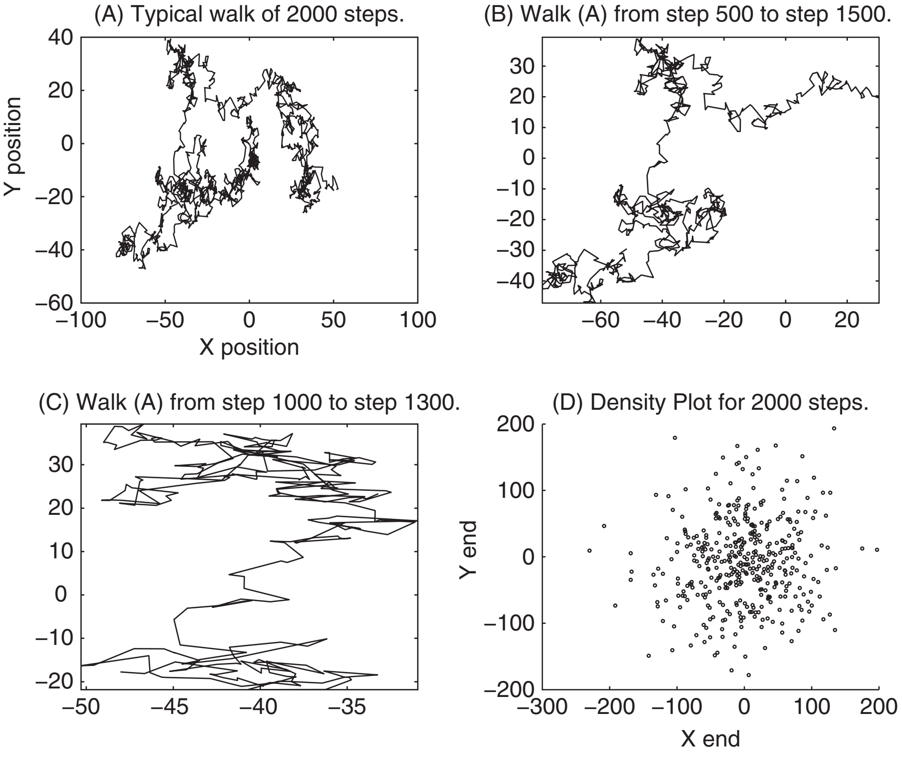


Fig. 5.3. Two-dimensional walk of 2000 steps.

that Brownian motion is scale-invariant, that is, the path of a particle observed under magnification has exactly the same characteristics no matter the magnification.

Specifically, scale invariance means that $\alpha^{-1}W_{\alpha^2 t}$ is also a Wiener process for any $\alpha \neq 0$. Since the steps sizes are normally distributed and proportional to times between observations, it follows that by dividing the step sizes by a fixed constant, say α, and simultaneously multiplying the variance by α^2, we will again have the same Wiener process.

Recurrence

As in one dimension, the 2-dimensional symmetric random walk on a lattice is recurrent and will return to the origin with certainty. Hence the walk will return to or hit the origin infinitely often. As a corollary of this, the walk will hit any given lattice point in the plane, and do so infinitely often. This is because from the origin, reaching any given point has a positive probability. Hence given infinitely many steps, eventually it will happen. The Markovian nature of the walk then ensures that it will return again and again.

Likewise, 2-dimensional Brownian motion is recurrent, however, not pointwise. In this case, this means that the particle will return to any *neighborhood* of the origin with probability 1, not necessarily exactly to the origin. For the same reason, the particle will eventually hit any given neighborhood in the plane.

However, in three dimensions and higher, recurrence does not hold. In these dimensions there is a positive probability that the walk, or the particle, will wander off to infinity and thus never return.

5.3 Random Walk Applications I

The foregoing developments have found their way into many applications. In this section we consider two of them.

In his remarkable Ph.D. thesis of 1900, Louis Batchelier introduced three of the fundamental concepts of modern finance: the invention of options as a mechanism for ameliorating risk, the efficient market hypothesis, and the use of Brownian motion to correctly price options. Batchelier's thesis was far ahead of its time and was forgotten until it was rediscovered in 1954. Our first application shows how Batchelier's ideas have been resurrected and refined and how the random walk is at the heart of option pricing, so important in today's financial markets.

Next we offer a brief description of self-avoiding walks. A simple requirement added to the standard random walk on a lattice leads to some very difficult and as yet unsolved mathematical questions that at the same time are important to modern chemistry and biology. The requirement is that the walk must not intersect itself; that is, no lattice point may be incorporated into the walk more than once. What makes questions about self-avoiding walks so hard is that the walk is not Markovian: the next step of the walk necessarily depends on the entire history of the walk. We give a background to the open questions in this area.

5.3.1 Options Pricing in Finance

A *put option* is a contract between the buyer or holder of the option and the seller or maker that allows the holder to sell, or "put," stock to the maker for a certain fixed price on a certain date in the future.

For example, suppose the price of General Electric stock is $32 per share right now. A company holding GE stock might be ruined if the price falls below $25 per share three months from now. So the company might buy a put contract guaranteeing that they will be able to sell their stock for $25 per share at that time; $25 is called the *strike price*. The contract is like an insurance policy: if the market price stays the same or increases, then the contract expires without being invoked. Otherwise, if the market price falls below the strike price on the date of expiration three months from now, then the company invokes the contract and limits their loss.

Of course, the maker of the option is taking a risk; the maker may have to buy the stock at $25 per share when the market price is less than that, maybe $20 per share, maybe even less. In exchange for taking on this risk, the maker charges a price called the *premium* for making the contract. If the contract is not invoked, the maker keeps the premium as a profit.

A *call option* is dual to the put option. The holder has the right to buy or "call" stock from the maker at the strike price on the date of expiration. For example, suppose Genentech trades for $79 per share today, but they have been working on a new product that could win FDA approval in the near future. If this happens, their stock price will soar. To benefit from this possibility a speculator buys call options guaranteeing the right to buy a certain number of shares of Genentech four months from now at a price of $85 per share. If the approval comes through and Genentech climbs to $92 per share at expiration, the speculator makes a $7 profit per share. If things go wrong and Genentech stays the same or declines, then the option contract expires worthless.

In the case of this call option, the maker runs the risk of having to buy stock at a market price above the strike price of $85, for example at $92, but sell it at the strike in order to fulfill the call contract. In consideration of this risk the maker charges a price for the option.

But what is a fair price for an option contract? This question, which was "in the air" in the 1960s and early 1970s, was solved in the form we see it today by Fischer Black and Myron Scholes in 1973 (and independently by Robert Merton in a separate publication that same year). The solution is based on the other two observations of Batchelier noted above. What is today called the *efficient market hypothesis* was rediscovered, developed, and put into clear form by Eugene Fama in his PhD thesis in the early 1960s. In an efficient market, prices on traded assets, such as stocks, already reflect all known information, and therefore their price movement over a short interval of time is completely random. To be sure, small price movements are more likely than large ones. By assuming that the step sizes are normally distributed, we see that the movement of a stock's price is a one-dimensional Brownian motion. The step size of the movement is small for stocks with a low price and high for stocks with a high price, that is, the step size is proportional to price. This refinement is called a *geometric Brownian motion* to reflect that fact. Notice that it is not called *geometric* as a result of it being a geometric phenomenon (that is, geometry as in lines and planes and circles), but that the variance of the process scales in a geometric fashion (proportional to the current value).

By exploiting Itō's stochastic calculus for random variables, Black and Scholes developed formulas for calculating fair option premiums on the assumption of geometric Brownian motion of the underlying stock. But we can calculate such premiums directly by simulating the geometric random walk of the stock's price.

Let S be the price of the stock on which the option contract is made. Let dt be a small unit of time, for example one day, and let dS be the change in price over that period of time. By the Brownian motion assumption, the fundamental equation for the stock price is

$$\frac{dS}{S} = \mu\, dt + \sigma\, dW, \tag{5.4}$$

where dW is the Brownian motion random variable and μ and σ are parameters particular to the stock itself. If there were no randomness in the stock's price, $dW = 0$, then the stock price would grow exponentially according to

$$\frac{dS}{dt} = \mu S.$$

Hence we see that μ is the growth (or decline) rate of the stock's price as a result of the growth of the company itself. As we will see shortly, μ plays no role in figuring option prices.

By contrast, the parameter σ is a very important constituent in option pricing. It is called *volatility* and is a reflection of the fact that some stocks' prices fluctuate more rapidly and with greater swings than others'. It is, in fact, the standard deviation of the prices of the particular stock underlying the option contract and can be approximated from the stock's price history.

Simulating a Geometric Random Walk

In order to simulate solutions to (5.4), we draw samples from a normal distribution for dW, use the equation to calculate dS, and carry the price S one time step forward, $S_{\text{new}} = S_{\text{old}} + dS$. The mean of dW is zero, but the variance must depend on the size of dt; in fact, Itō showed that the variance of dW must be taken as dt. The equation can therefore be rewritten as

$$\frac{dS}{S} = \mu\, dt + \sigma\sqrt{dt}N, \tag{5.5}$$

where now $N \sim N(0,1)$ is normally distribution with mean 0 and variance 1.

As to dimensions, every term in (5.5) is dimensionless. Therefore if dt is in days, μ is in per day and σ is in per square root day. But these parameters are usually given per year; hence we take dt in years as well. Then μ is in per year and volatility is in per square root year; however, the latter term is never used in practice; instead, the term per year is used for it as well.

Risk-Neutral Valuation

We said that the option price must be fair, but what does that mean? In fact, the fairness principle has far-reaching implications. For example, it means that no one can make "free" money. By investing one's money in a guaranteed bank account paying an interest rate, one is earning riskless money. But this is not free money because the bank's use of money has value; the interest paid is compensation for its use over time. Let r be the best *riskless interest rate* possible; this is usually on government bonds.

An implication of the principle of fairness is that μ in (5.5) must be taken equal to r in the computation of a fair option price. This is known as *risk-neutral evaluation*. If μ were larger and this earning rate guaranteed, then one would invest in this stock instead of government bonds and μ would be considered as the risk-free rate. Hence a value of $\mu > r$ is the result of risk and, if used in (5.5), would give rise to an unfair option price. In particular, it would not be fair to discount future money at the rate of r, see (5.6), while allowing a drift at the rate of μ. For more on this see [Hul03].

Calculating the Price of an Option

Here then is how to figure the price of an option by simulation. Select the type of option, for example a call option, the desired time period, for example 90 days, and from the underlying stock note its current price, for example $S_0 = \$20$, and volatility, for example $\sigma = 60\%$ per year. And as discussed above, note the current risk-free interest rate, for example 3.10% per year, from `www.federalreserve.gov`.

```
  Matlab
> nDays = 90;
> dt = 1/365.0; % use time in years
> T = nDays*dt; % expiry in years
> S0=20; % starting stock price
> X=25; % strike price
> r = .031; % risk-free rate
> sigma = .6; % volatility
> expTerm = r*dt;
> stddev = sigma*sqrt(dt);
> nTrials = 10000;
> value=0;
> for j=1:nTrials
    n = randn(1,nDays); % vector of N(0,1) random numbers
    S = S0;
    for i=1:nDays
      dS = S*(expTerm + stddev*n(i));
      S = S + dS;
    end
    S90(j) = S; % to histogram
    value = value + max(S-X,0);
  end
> value = value/nTrials; % expected value
> Price = exp(-r*T)*value % discount back
> hist(S90,0:.5:65)
```

Each simulation of (5.5) is a possible future path of the stock's price over the 90 days. One cannot say which future will materialize, but we can histogram a large number of them and know their distribution. Figure 5.4 shows the histogram of nTrials = 10,000 such paths.

Notice that the final price, S_{90}, ranges from \$6 to over \$60. This is not a normal distribution because the random walk is geometric; the price can never fall below zero. Instead, it is $\log S$ that is normal; S itself is called *lognormal*; an asymmetric distribution.

Next we must select the strike price X of our option, for example $X = \$25$. We then calculate the expected value of the option at expiration. If the ending stock price is less than the strike price, the call option expires worthless. If the ending stock price is \$26, then the op-

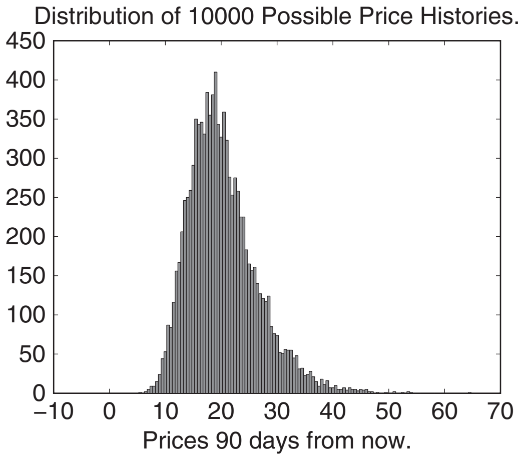


Fig. 5.4. Possible price histories over 90 days.

tion contract states that we can buy the stock at $25, so the value of the option is $1. In general, the value of the option at expiration is

$$\max(S_T - X, 0),$$

where S_T is the stock price at expiration. Hence an estimate of the expected value of the option at expiration is given by

$$E = \frac{1}{\text{nTrials}} \sum_{i=1}^{\text{nTrials}} \max(S_{90}(i) - X, 0).$$

This is the option's value at expiration, day 90. To find the value on day 0 we must discount the 90-day value back 90 days at the risk-free interest rate,

$$P = e^{-rT} E. \tag{5.6}$$

The exact value for this call option given by the Black–Scholes formula is 0.9056. In running the simulation above six times, the outcomes were 0.9388, 0.8581, 0.8943, 0.8699, and 0.9085. These runs have a mean of 0.8898 and a standard deviation of 0.0276, so the result is within 0.6 standard deviation of the exact value. As usual, running more trials will reduce the error at the rate of one over the square root of the number of trials.

5.3.2 Self-Avoiding Walks

A *self-avoiding walk* (SAW) is a random walk on a cubical (or hexagonal) lattice starting from the origin with the property that no site is revisited by the walk. An SAW in one dimension is trivial; in 2 or 3 dimensions even the most basic questions about them are still open. These include (1) how many different SAWs are there of a given number of steps? and (2) what is the average displacement of an n-step SAW? Here *displacement* refers to the geometric straight-line distance between the terminal point and the starting point of the walk.

Letting c_n denote the number of n-step walks, for a 2-dimensional cubical (i.e., square) lattice it is easy to count directly that $c_1 = 4$, $c_2 = 12$, $c_3 = 36$, and $c_4 = 100$. It is conjectured that in general,

$$c_n \approx 2.64^n.$$

As to the displacement of an n-step walk, for square lattices the conjecture is that the expected displacement squared is

$$\mathbb{E}(\text{displacement}^2) \approx n^{3/2}.$$

Besides a purely mathematical interest in SAWs, their study is very important in the fields of industrial polymer chemistry and biological macromolecules. For example, industrial polymers are made up of small units called *monomers* chemically linked together, sometimes by the thousands. Abstractly, the lattice points of an SAW play the role of the monomers, while the edges play the role of the links. Of course no two monomers can occupy the same location in space and hence the need to impose the self-avoiding condition.

The two basic properties about SAWs mentioned above, c_n and displacement, imply physical properties of the polymers they model.

In coding an SAW, the history of the walk must be kept and consulted for each new step. The most efficient way to obtain long walks is to choose the next direction, north, south, east, west (in two dimensions), from among those avoiding immediate intersection. If from the current location of the walk, sites north and west are occupied, the random choice is restricted to east and south. It is possible that the walk can self-trap, in which case there are no available sites and the walk must end. At this point, the statistical information of the walk is recorded, the history is cleared out, and a new walk is started.

Problem 15 involves using simulation to estimate c_n and walk displacement for a self-avoiding walk on a square lattice in the plane.

5.4 Gambler's Ruin

The adaptation of the simple random walk to gambling is an endeavor to understand the rise and fall of a gambler's fortune over time. In fact, probability owes its origins to gambling, stemming, as it were, from letters exchanged by Pierre Fermat and Blaise Pascal about gambling problems.

Suppose, as we did in the introduction, a gambler bets 1 unit, e.g., 1 dollar, against the house with probability p of winning the bet and probability $q = 1 - p$ of losing it. Assume that the gambler starts with x units of money. Let $X(t)$ be the random variable equal to the gambler's fortune at time t; hence $X(0) = x$. After one play of the game, the gambler's fortune will have increased or decreased by one unit, so $X(1) = x + 1$ with probability p and $X(1) = x - 1$ with probability q; now continue. We see that this is a random walk just as in Section 5.1, except that the walk starts at x instead of 0.

Well almost; the walk cannot go below 0 in this case, for then the gambler will be ruined. Likewise, the house also has finite capital; assume that it starts the game with h units. So all the money in the game is the constant $a = x + h$. At any time, the house's fortune is the difference between this and the gambler's fortune, $H(t) = a - X(t)$. If $X(t) = a$, then the house is broken and again play stops.

Play continues so long as $0 < X(t) < a$ and stops the first time one of these barriers is reached. An important question is whether play will go on indefinitely. The answer is no; given enough time, either the gambler will be ruined or the gambler will break the house. (At least, with probability one this will happen; all such statements are necessarily probabilistic statements.) We will show that this is so momentarily.

It is easy to see that if the game is fair, $p = 1/2$, then the expectation on every play is the original gambler's fortune. For example,

$$E(X(1)) = \frac{1}{2}(x+1) + \frac{1}{2}(x-1) = x$$

and

$$E(X(2)) = \left(\frac{1}{2}\right)^2 (x+2) + 2\frac{1}{2}\cdot\frac{1}{2}x + \left(\frac{1}{2}\right)^2 (x-2) = x,$$

and so on. Processes having this expectation-preserving property are called *martingales*. Further, we know from Section 5.1 that the walk stochastically wanders away from the starting point in proportion to

$\sqrt{t}$. This gives an indication as to how long the game is expected to last before the gambler is ruined or the house is broken.

Play will end at an absorbing barrier, but which one? Let r_x be the probability that the gambler will be ruined if his starting capital is x. Similarly, let h_x be the probability that the gambler breaks the house. If we can show that $r_x + h_x = 1$, then it follows that the game cannot go on indefinitely, as asserted above.

To determine r_x and h_x decompose on one play of the game. If the gambler loses the first game, the chance of ruin is the same as if he starts with $x-1$ units. Similarly, if he wins the first game, the chance of ruin is the same as if his starting capital is $x+1$ units. Hence

$$r_x = qr_{x-1} + pr_{x+1}, \quad x = 2, 3, \ldots, a-2. \tag{5.7}$$

The gambler starting with 1 unit, $x = 1$, or all the money except one unit, $x = a-1$, are special cases. In the first case, a loss gives ruin immediately, so

$$r_1 = q + pr_2.$$

In the second case, a win breaks the house, so the subsequent chance of ruin is 0. Thus

$$r_{a-1} = qr_{a-2}.$$

But by putting $r_0 = 1$ and $r_a = 0$, these special cases follow from (5.7).

Before going on, notice that the equations for h_x are just the same except at the boundary points, where we have instead $h_0 = 0$ and $h_a = 1$.

Equation (5.7) is an example of a difference equation. The theory of linear constant-coefficient difference equations closely tracks that of linear constant-coefficient differential equations (for an introduction to the theory of difference equations, see [Gri03]). One tries a solution of the form $r_x = b^x$ for some constant b. Substituting this into (5.7) yields

$$b^{x-1}\left(-q + b - pb^2\right) = 0.$$

Solving the quadratic for b, and remembering that $q = 1-p$, gives

$$b = \frac{1 \pm \sqrt{1-4pq}}{2p} = \frac{1 \pm (1-2p)}{2p}.$$

Hence if $p \neq 1/2$, the two distinct solutions are $r_x = 1$ and $r_x = (q/p)^x$. In this case the general solution is

$$r_x = A + B(q/p)^x$$

for constants A and B such that $r_0 = 1$ and $r_a = 0$. Solving for these gives

$$r_x = \frac{(q/p)^a - (q/p)^x}{(q/p)^a - 1}. \tag{5.8}$$

In case $p = q = 1/2$, the repeated root is $b = 1$ and the two independent solutions are $r_x = 1$ and $r_x = x$. The general solution is $r_x = A + Bx$, and after satisfying the boundary conditions, we get

$$r_x = 1 - \frac{x}{a}. \tag{5.9}$$

Repeating the above for h_x, we find that

$$h_x = \frac{(q/p)^x - 1}{(q/p)^a - 1}, \quad p \neq 1/2. \tag{5.10}$$

And in a fair game we get

$$h_x = \frac{x}{a}, \quad p = 1/2. \tag{5.11}$$

Therefore $r_x + h_x = 1$ for all x, as expected. Recall that this means that for any amount x that the gambler starts with, the probability that either the gambler is ruined or that the house is broken is 1. Thus one of these two events is sure to happen.

Martingale Solution

A stochastic process that can be interpreted as a player's fortune in a fair game is called a *martingale*. The random walk above with $p = 1/2$ is an example of a martingale.

If the random walk is a fair game and r_x is the probability of ruin as before, then a player's final expected fortune is

$$r_x \cdot 0 + (1 - r_x) \cdot a.$$

Since the initial fortune is x, we equate these two to get the equation

$$x = (1 - r_x)a,$$

or $r_x = 1 - x/a$ as before.

Some Consequences of the Ruin Equations

Normally, a gambling casino has much more money than a gambler, and therefore the probability of breaking the house, even in a fair game, is very small. For example, suppose the player starts with \$10,000 and meanwhile the casino has \$1,000,000. Then according to (5.11), the chance of breaking the house is

$$\frac{10{,}000}{1{,}000{,}000 + 10{,}000} = 0.0099,$$

about 1%. And so the chance of the gambler's ruin is $1 - 0.0099 = 0.9901$. But the games are never fair. Playing red in roulette with two green slots on the wheel has a winning probability of $p = 18/38 = 0.474$. Hence $q = 1 - p = 0.526$, and from (5.10) the chance of breaking the house is

$$\frac{(q/p)^x - 1}{(q/p)^a - 1} = \frac{(0.526/0.474)^{10000} - 1}{(0.526/0.474)^{1010000} - 1} \approx 3.96 \times 10^{-45208},$$

none! But suppose there were a game for which $p = 0.4999$, within one ten-thousandth of being fair. What is the chance of breaking the house then? Again from (5.10) it is about 1.89×10^{-174}, tiny. As one can see, even the most minute shift from a fair game has extreme consequences, though of course, one usually doesn't try to break the house!

Expected Duration of the Game

As noted above, the dispersion rate of $\sqrt{t}$ gives an indication as to how long a game can last on average. But the exact value is not hard to derive. Let $T(x)$ be the expected time until absorption from an initial fortune of x and assume that this is a fair game. Just as above, by decomposing on one play, we have

$$T(x) = 1 + \frac{1}{2}T(x+1) + \frac{1}{2}T(x-1), \quad 0 < x < a.$$

Of course $T(0) = T(a) = 0$.

This time, the difference equation is inhomogeneous; rearranging terms above gives

$$T(x) - \frac{1}{2}T(x+1) - \frac{1}{2}T(x-1) = 1.$$

Obtain the homogeneous solution, with right-hand side equal to 0, as above, by trying $T(x) = b^x$. The repeated roots $b = 1$ mean that T has the form $T(x) = A + Bx$ for some constants A and B. The form of the right-hand side cannot be constant or constant times x, since these are homogeneous solutions. Hence the right-hand side must be of the form $T(x) = Cx^2$. Substitution gives

$$Cx^2 - \frac{C}{2}(x+1)^2 - \frac{C}{2}(x-1)^2 = 1,$$

from which it follows that $C = -1$. Therefore the complete solution is

$$T(x) = A + Bx - x^2.$$

Now knowing that $T(0) = T(a) = 0$ gives

$$T(x) = x(a - x) \tag{5.12}$$

as the expected duration of the game.

5.4.1 Gambling Schemes

Let us suppose the gambler brings \$$x$ to the casino and decides to play until either being ruined or winning a fixed amount, say \$$Q$. To be definite, assume that the gambler starts with \$1023 and wants to win just one dollar.

Moreover, suppose the gambler bets just \$1 per play with probability of winning equal to p. In this case we can use the ruin equations with the assumption that the house has the amount $h = Q = \$1$, so that $a = Q + x = \$1024$ in those equations.

Again using (5.10), the gambler succeeds with probability

$$\frac{(q/p)^x - 1}{(q/p)^a - 1} = \frac{(0.526/0.474)^{1023} - 1}{(0.526/0.474)^{1024} - 1} = 0.90.$$

Of course with probability 0.10, or 10% of the time the gambler loses all \$1023. Hence the gambler's expectation is

$$\$1 \times 0.90 - \$1023 \times 0.10 = -\$101.40.$$

Perhaps a consolation is that it should take a long time before losing everything, since from (5.12) we have

$$T(x) = x(a - x) = 1023(1) = 1023;$$

that is, 1023 plays before ruin or success on average, a few days' worth, one would think. It is interesting to note that since the gambler starts with \$1023 and bets only \$1 each play, it takes at least 1023 plays before the gambler is ruined.

This is the outcome when the gambler uses the strategy of betting \$1 each play. Maybe a different strategy is better. For example, consider the "doubling-up" strategy. The gambler decides in advance to win exactly 1 unit. If she wins on the first play, she goes home having achieved her goal. If she loses on the first play, she plays again, this time doubling her previous bet. Hence on the second play she bets 2 units. If she wins this time, the payoff is 2 units. This covers her 1-unit loss on the first game and achieves the predetermined 1-unit goal. So she now quits. Otherwise, she continues, doubling her bet, until she wins the first time. If the gambler were able to play long enough, eventually she would succeed. But two facts intervene: the gambler has limited resources, for example \$1023, and the house imposes betting limits.

In the doubling-up strategy, the gambler bets $\$2^{n-1}$ on the nth play if she has lost on the previous $n-1$ plays. The amount bet in total up to and including the nth bet is $1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1$. Since $2^{10} - 1 = 1023$, the gambler can survive at most nine losses in a row; the tenth will leave her broke. Since the probability of ten losses in a row is $0.526^{10} = 0.0.00162$, the expectation, or average rate of loss, with this strategy is

$$\$1 \times (1 - 0.00162) - \$1023 \times 0.00162 = -\$0.660,$$

a much better result. Furthermore, the probability of success is about 99.84%, much better than 90%. This shows dramatically that bold play is much preferable.

If fortunes are bounded, then there can be no gambling scheme guaranteed to win. If the games are fair, then the expectation of the gambler's fortune at any time is equal to her starting fortune for any manner of betting, and so again there are no guaranteed winning schemes!

5.5 Random Walk Applications II—Kelly's Criterion in Finance

Working at Bell Labs in the early 1950s, John Kelly considered the problem of deciding how much money to risk on probabilistic outcomes that are nonetheless in your favor (see [Kel56]). His motivation was speculation about gambling tips over noisy communication channels, but his criterion has been much more widely interpreted. Among others, it is the basis of a strategy for asset allocation in the stock market.

5.5.1 The Simple Kelly Game

As a simple example, what fraction f of one's bankroll should be bet on a coin flip that has p chance of winning if the gain per unit bet is γ? The expected payoff of a game is

$$E = \gamma p - q, \tag{5.13}$$

where $q = 1 - p$ is the probability of losing. Since we are assuming that the bet is in our favor, $E > 0$.

The other assumption Kelly made is that the game is to be played over and over indefinitely. Hence the key insight: *we should strive to maximize our expected growth rate g*. Let X_N be our fortune after the Nth play; X_0 is our initial bankroll. The growth rate satisfies the equation $e^{gN} X_0 = X_N$, and therefore

$$g = \frac{1}{N} \log \frac{X_N}{X_0}. \tag{5.14}$$

Note that g is a random variable, since X_N is. After one play we have

$$\begin{aligned} X_1 &= \begin{cases} \text{if win} & X_0 + \gamma f X_0 = (1+\gamma f)X_0, \\ \text{if lose} & X_0 - f X_0 = (1-f)X_0, \end{cases} \\ &= Q_1 X_0, \end{aligned}$$

where Q_1 is the random variable

$$Q_1 = \begin{cases} 1+\gamma f, & \text{with probability } p, \\ 1-f, & \text{with probability } q. \end{cases}$$

After two plays $X_2 = Q_2 Q_1 X_0 = Q^2 X_0$, since $Q_2 = Q_1 = Q$, and after n plays we have $X_N = Q^N X_0$, so

$$g = \frac{1}{N} \log Q^N = \log Q.$$

The expectation of g is

$$E(g) = E(\log Q) = p \log(1+\gamma f) + q \log(1-f).$$

Maximize with respect to f by setting the derivative to zero,

$$0 = \frac{p\gamma}{1+\gamma f} - \frac{q}{1-f} = \frac{p\gamma(1-f) - q(1+\gamma f)}{(1+\gamma f)(1-f)},$$

for a solution of

$$f = \frac{\gamma p - q}{\gamma} = \frac{E}{\gamma}. \tag{5.15}$$

Note the simple form of the answer, the expectation over the gain, or sometimes referred to in brief as the edge over the odds. This is *Kelly's criterion*.

As an example, let the game be a weighted coin flip with probability $p = 0.6$ of winning and even-money payoff; hence $\gamma = 1$. Then the optimal betting fraction is $f = 0.6 - 0.4 = 0.2$. In Figure 5.5 we show $N = 300$ plays of this game for various fractions f; note that the y-axis scales are different. The runs were obtained by simulation using the following code, and therefore each is only a single possible fortune history. However, some characteristics are general; for example, as f increases from 0.05 to 0.25 the ending fortunes increase, but at the same time the fortune is subject to wilder oscillations as well. In Figure 5.6 a long-term $N = 20{,}000$ comparison is shown between $f = 0.2$ and $f = 0.25$.

```
 Matlab
> function fitnesses = kellyEval(popSize,chromPop);
> popSize = 1; % remove for use in GA
> chromPop=0.05; % remove for use in GA
> F=linspace(1,1,popSize); % initial fortune
> N=1000; % iterations of the game
> h=linspace(1,1,N); % histogram, remove for GA
> for i=2:N
>  w = F.*chromPop; %term-by-term wager
>  % play 60/40 game
>  r = (rand(1,popSize) < 0.6); % 1 if <0.6 else 0
>  r = 2*r-1; % +1 if <0.6 else -1
>  w = r.*w; % +bet if <0.6, else -bet
>  h(i)=F; % remove for use in GA
>  F = F+w; % new fortune
> end
> fitnesses = exp((1/N)*log(F));
```

5.5.2 The Simple Game with Catastrophic Loss

As a step closer to financial applications, we modify the simple game to include the possibility of a large loss. Consider bets on the outcome of tossing two coins. Let event O_1 be *the two coins are different*, let O_2 be the event *two heads*, and O_3 the event *two tails*. Then $p_1 = 1/2$, $p_2 = p_3 = 1/4$, and so fair payoff odds are $1 : 1$ for O_1 and $3 : 1$ for O_2 and O_3. That is, if we bet \$1 on O_1 and win, then we get our \$1 back and \$1 more as the gain.

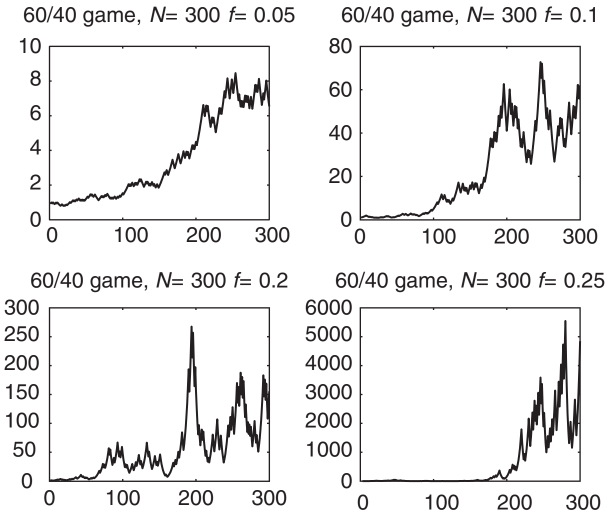


Fig. 5.5. 60/40 game fortunes for various betting fractions.

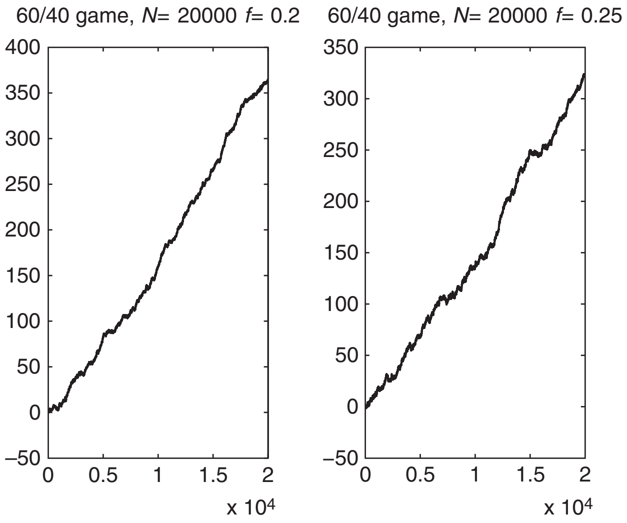


Fig. 5.6. logarithm of 60/40 game fortunes for $f = 0.2$ vs $f = 0.25$

As in all Kelly examples, we play only if the payoff odds are in our favor, so for the purposes of this example assume that for a \$1 bet on O_1, we make \$4 if it occurs, we lose the bet if O_2 occurs, and we lose our bet and \$3 more if outcome O_3 occurs. Let γ denote the amount we gain upon a win per unit bet and let λ denote the amount we lose upon a loss per unit bet. Thus $\gamma_1 = 4$, $\lambda_2 = 1$, and $\lambda_3 = 4$. The expected gain is

$$E = p_1\gamma_1 - p_2\lambda_2 - p_3\lambda_3 = \left(\frac{1}{2}\right)(4) - \left(\frac{1}{4}\right)(1) - \left(\frac{1}{4}\right)(4) = \frac{3}{4}.$$

Because of λ_3, a bet of more than one-fourth our stake is prohibited, since we cannot cover the loss in this case. Moreover, a bet equal to one-fourth our stake loses everything eventually and therefore will not lead to maximum growth, so this too is not the solution. Note that in this example, although the bet is \$1, the risk is as high as four times the bet. This represents the catastrophic loss.

Again let f denote the fraction of our fortune bet per play. We have

$$\begin{aligned} X_1 &= \begin{cases} \text{if } O_1 & X_0 + fX_0\gamma_1 = (1+f\gamma_1)X_0, \\ \text{if } O_2 & X_0 - fX_0\lambda_2 = (1-f\lambda_2)X_0, \\ \text{if } O_3 & X_0 - fX_0\lambda_3 = (1-f\lambda_3)X_0, \end{cases} \\ &= QX_0, \end{aligned}$$

where Q is the random variable

$$Q = \begin{cases} 1+f\gamma_1, & \text{with probability } p_1, \\ 1-f\lambda_2, & \text{with probability } p_2, \\ 1-f\lambda_3, & \text{with probability } p_3. \end{cases}$$

As before, after n plays the expected growth rate is

$$E(g) = E(\log Q) = p_1\log(1+f\gamma_1) + p_2\log(1-f\lambda_2) + p_3\log(1-f\lambda_3).$$

The maximum is given by

$$0 = E'(g) = \frac{p_1\gamma_1}{1+f\gamma_1} - \frac{p_2\lambda_2}{1-f\lambda_2} - \frac{p_3\lambda_3}{1-f\lambda_3}.$$

Combining these fractions gives a numerator of

$$0 = E - f\left(p_1\gamma_1(\lambda_2+\lambda_3) + p_2\lambda_2(\gamma_1-\lambda_3) + p_3\lambda_3(\gamma_1-\lambda_2)\right) + \gamma_1\lambda_2\lambda_3 f^2. \tag{5.16}$$

Since $f < \frac{1}{4}$, assume that the last term is small and temporarily ignore it; thus approximately,

$$\begin{aligned} f &= \frac{E}{p_1\gamma_1(\lambda_2+\lambda_3)+p_2\lambda_2(\gamma_1-\lambda_3)+p_3\lambda_3(\gamma_1-\lambda_2)} \\ &= \frac{3/4}{13} = 0.0577. \end{aligned}$$

By including the quadratic term, the exact solution is

$$f = 0.0625.$$

As one can see from this example, in more complicated asset allocation problems, solving for the optimizing fraction or fractions leads to nonlinear systems of equations that are difficult to solve. But since these are results about stochastic situations, we should be able to find the answers, no matter how complicated, by Monte Carlo simulation. Let us now consider an actual stock market problem.

5.5.3 Option Trading Application I

In many types of option contracts, the contract either makes a certain amount of money or it expires worthless and one's premium, the cost of the option, is lost. Further, the payoff is finalized on a specific date. Hence this is a perfect setup for the application of Kelly's criterion. In the following, we apply the criterion to the reported trading experience of a seasoned trader who kept careful records.

A certain option trader making *credit spreads*[1] reports these statistics over the last 75 trades: the number of trades that gained money, 66; 9 lost money; the average gain per trade, \$659.12; the maximum loss, \$2837.00; the average loss excluding the maximum, \$1669.32. Let's apply what we have learned above to figure the fraction of the trader's

[1] A credit spread is the option strategy of selling an option on a stock at one price and buying a lower-priced option on the same stock with the same expiration date. The cost of a call option goes down as the strike price goes up. By selling the option with the lower strike and buying the one with the higher strike, the trader brings in net income. For example, suppose IBM stock is at \$114 per share. A call option for IBM with a one-month expiration sells for \$3.67 if the strike price is \$115 and \$1.85 if it is \$120. Selling the 115 option and buying the 120 earns a net credit of \$3.67 − \$1.85 = \$1.82 per share. If IBM stays about the same or retreats, the \$1.82 is all profit. Why buy the 120 call? If IBM advances, say to \$125 (or more!), the trader must buy stock at this price per share and sell at \$115 to fulfill the 115 contract, a loss of \$10 per share. Buying the 120 call is insurance; invoking it to buy IBM at \$120 limits the loss to \$5 per share.

By duality, there is a comparable trade using put options in place of calls. (What is it?)

capital that should be invested in this type of trade to maximize expected growth.

The situation here is exactly like the two-coins game above. We take each "bet" to be the average loss, \$1,669.32; hence $\lambda_2 = 1$. Then $\gamma_1 = 659.12/1669.32 = 0.3948$ and the maximum loss per dollar is $\lambda_3 = 2837/1669.32 = 1.70$. Finally, the probabilities are reported as $p_1 = 66/75 = 0.880$, $p_2 = 8/75 = 0.107$, and $p_3 = 1/75 = 0.013$. The expected gain is

$$E = (0.88)(0.3948) - (0.107)(1) - (0.013)(1.70) = 0.2183.$$

To find the maximizing fraction, recall the quadratic (5.16)

$$0 = E - f(p_1\gamma_1(\lambda_2 + \lambda_3) + p_2\lambda_2(\gamma_1 - \lambda_3) + p_3\lambda_3(\gamma_1 - \lambda_2)) + \gamma_1\lambda_2\lambda_3 f^2.$$

Solving gives $f = 0.455$, or about 46%. (The linear approximation is not good here, coming in at 0.28.)

5.5.4 Option Trading Application II

Suppose we want to allocate an investment fund among several possible investments. To keep it simple, suppose there are just two: let option play A be the credit spread considered above and let option play B be a *calendar spread*[2] having the following experience. Assume that the statistics of the calendar spread are number of trades, 12; number of winning trades, 10; average gain per trade, \$555.20; maximum loss, \$124.00; average loss excluding the maximum, \$90. We take this latter figure to be the size of the bet. The γ's and λ's are

$$\gamma = \frac{555.20}{90} = 6.17, \quad \lambda_{\text{avg}} = 1, \quad \lambda_{\max} = 124/90 = 1.38.$$

The expected gain is then

$$E = \left(\frac{10}{12}\right)(6.17) - \left(\frac{1}{12}\right)(1) - \left(\frac{1}{12}\right)(1.38) = 4.943.$$

[2] A trade in which a near-term expiration option is sold and a longer term option on the same stock and for the same strike price is bought. For example, with IBM at \$114, the one-month call option with strike price \$115 is \$3.67, while the two-month call with the same strike sells for \$5.32. Selling the one-month call and buying the two-month call is for a net debit of \$5.32 – \$3.67 = \$1.65. But if IBM stays about the same over the next month, the one-month call will expire worthless and entail zero obligation. Meanwhile, the two-month call (now one month from expiration) should be worth about \$3.67. Altogether, a net \$3.67 – \$1.65 = \$2.02 profit.

Obviously this trader did unusually well!

If this were the only investment, we should risk the fraction given by the smallest root of the quadratic equation (5.16) copied from the two-coins game,

$$0 = E - f\left(p_1\gamma_1(\lambda_2+\lambda_3) + p_2\lambda_2(\gamma_1-\lambda_3) + p_3\lambda_3(\gamma_1-\lambda_2)\right) + \gamma_1\lambda_2\lambda_3 f^2,$$

whose root is $f = 0.625$.

But what if we make both investments? Thus allocate f_A of our capital for option play A and f_B for B. In the following, let p denote the success probability, q denote the average loss probability, and r the maximum loss probability. We assume that the success or failure of each option play is independent of the other. (This is probably unjustified; in Problem 19 we show how correlation may be taken into account.) Note that both A and B might suffer the maximum loss; hence we must always ensure that

$$\lambda_\mathrm{A} f_\mathrm{A} + \lambda_\mathrm{B} f_\mathrm{B} < 1.$$

There are now nine possibilities, as listed in Table 5.1.

Outcome	Probability	Q
A wins, B wins	$p_\mathrm{A}p_\mathrm{B}$	$1+f_\mathrm{A}\gamma_\mathrm{A}+f_\mathrm{B}\gamma_\mathrm{B}$
A wins, B loses	$p_\mathrm{A}q_\mathrm{B}$	$1+f_\mathrm{A}\gamma_\mathrm{A}-f_\mathrm{B}$
A wins, B maxes	$p_\mathrm{A}r_\mathrm{B}$	$1+f_\mathrm{A}\gamma_\mathrm{A}-f_\mathrm{B}\lambda_\mathrm{B}$
A loses, B wins	$q_\mathrm{A}p_\mathrm{B}$	$1-f_\mathrm{A}+f_\mathrm{B}\gamma_\mathrm{B}$
A loses, B loses	$q_\mathrm{A}q_\mathrm{B}$	$1-f_\mathrm{A}-f_\mathrm{B}$
A loses, B maxes	$q_\mathrm{A}r_\mathrm{B}$	$1-f_\mathrm{A}-f_\mathrm{B}\lambda_\mathrm{B}$
A maxes, B wins	$r_\mathrm{A}p_\mathrm{B}$	$1-f_\mathrm{A}\lambda_\mathrm{A}+f_\mathrm{B}\gamma_\mathrm{B}$
A maxes, B loses	$r_\mathrm{A}q_\mathrm{B}$	$1-f_\mathrm{A}\lambda_\mathrm{A}-f_\mathrm{B}$
A maxes, B maxes	$r_\mathrm{A}r_\mathrm{B}$	$1-f_\mathrm{A}\lambda_\mathrm{A}-f_\mathrm{B}\lambda_\mathrm{B}$

Table 5.1. Parameters for the two-option allocation data.

Differentiate the expected growth rate equation first with respect to f_A,

$$\begin{aligned}0 = {} & \frac{p_\mathrm{A}p_\mathrm{B}\gamma_\mathrm{A}}{(1+\gamma_\mathrm{A}f_\mathrm{A}+\gamma_\mathrm{B}f_\mathrm{B})} + \frac{p_\mathrm{A}q_\mathrm{B}\gamma_\mathrm{A}}{(1+\gamma_\mathrm{A}f_\mathrm{A}-f_\mathrm{B})} + \frac{p_\mathrm{A}r_\mathrm{B}\gamma_\mathrm{A}}{(1+\gamma_\mathrm{A}f_\mathrm{A}-\lambda_\mathrm{B}f_\mathrm{B})} \\ & - \frac{q_\mathrm{A}p_\mathrm{B}}{(1-f_\mathrm{A}+\gamma_\mathrm{B}f_\mathrm{B})} - \frac{q_\mathrm{A}q_\mathrm{B}}{(1-f_\mathrm{A}-f_\mathrm{B})} - \frac{q_\mathrm{A}r_\mathrm{B}}{(1-f_\mathrm{A}-\lambda_\mathrm{B}f_\mathrm{B})} \\ & - \frac{r_\mathrm{A}p_\mathrm{B}\lambda_\mathrm{A}}{(1-\lambda_\mathrm{A}f_\mathrm{A}+\gamma_\mathrm{B}f_\mathrm{B})} - \frac{r_\mathrm{A}q_\mathrm{B}\lambda_\mathrm{A}}{(1-\lambda_\mathrm{A}f_\mathrm{A}-f_\mathrm{B})} - \frac{r_\mathrm{A}r_\mathrm{B}\lambda_\mathrm{A}}{(1-\lambda_\mathrm{A}f_\mathrm{A}-\lambda_\mathrm{B}f_\mathrm{B})}\end{aligned}$$

and then with respect to f_{B},

$$\begin{aligned}0 = &\frac{p_{\mathrm{A}}p_{\mathrm{B}}\gamma_{\mathrm{B}}}{(1+\gamma_{\mathrm{A}}f_{\mathrm{A}}+\gamma_{\mathrm{B}}f_{\mathrm{B}})} - \frac{p_{\mathrm{A}}q_{\mathrm{B}}}{(1+\gamma_{\mathrm{A}}f_{\mathrm{A}}-f_{\mathrm{B}})} - \frac{p_{\mathrm{A}}r_{\mathrm{B}}\lambda_{\mathrm{B}}}{(1+\gamma_{\mathrm{A}}f_{\mathrm{A}}-\lambda_{\mathrm{B}}f_{\mathrm{B}})}\\ &+ \frac{q_{\mathrm{A}}p_{\mathrm{B}}\gamma_{\mathrm{B}}}{(1-f_{\mathrm{A}}+\gamma_{\mathrm{B}}f_{\mathrm{B}})} - \frac{q_{\mathrm{A}}q_{\mathrm{B}}}{(1-f_{\mathrm{A}}-f_{\mathrm{B}})} - \frac{q_{\mathrm{A}}r_{\mathrm{B}}\lambda_{\mathrm{B}}}{(1-f_{\mathrm{A}}-\lambda_{\mathrm{B}}f_{\mathrm{B}})}\\ &+ \frac{r_{\mathrm{A}}p_{\mathrm{B}}\gamma_{\mathrm{B}}}{(1-\lambda_{\mathrm{A}}f_{\mathrm{A}}+\gamma_{\mathrm{B}}f_{\mathrm{B}})} - \frac{r_{\mathrm{A}}q_{\mathrm{B}}}{(1-\lambda_{\mathrm{A}}f_{\mathrm{A}}-f_{\mathrm{B}})} - \frac{r_{\mathrm{A}}r_{\mathrm{B}}\lambda_{\mathrm{B}}}{(1-\lambda_{\mathrm{A}}f_{\mathrm{A}}-\lambda_{\mathrm{B}}f_{\mathrm{B}})}.\end{aligned}$$

This is a difficult nonlinear system to solve by most methods. However, it may readily be solved by genetic algorithms.

The fitness component of the GA is a random walk simulation of each play using the probabilities p_{A}, …, r_{B} as in Table 5.1.

The solution obtained by the GA is

$$f_{\mathrm{A}} = 0.073 \text{ and } f_{\mathrm{B}} = 0.619.$$

Note that the maximum loss amount is

$$\lambda_{\mathrm{A}}f_{\mathrm{A}} + \lambda_{\mathrm{B}}f_{\mathrm{B}} = 1.7 \times 0.073 + 1.38 \times 0.619 = 0.978,$$

or nearly the entire capital. The probability that this happens is $0.0133 \times 0.083 = 0.0011$.

Here is the GA Algorithm 1 code, page 147, adapted to this problem. Notice that it puts together many of the techniques that we have seen earlier such as selecting with breakpoints and permutations. The function kellyEval called here is given on page 188.

Matlab

```
> % a "chromosome" here is a fraction between 0 and 1
> % a population is a vector of chromosomes
> popSize=32;
> chromPop= rand(1,popSize); %initalize randomly
>  % space for their fitnesses = exp(growth rate)
> fitnesses=linspace(0,0,popSize);
> newPop=linspace(0,0,popSize);
> upper = linspace(1,1,popSize); % keep fractions < 1
> lower = linspace(0,0,popSize); % keep fractions > 0
> bp = linspace(0,0,popSize); % roulette wheel breakpts
> pm = 0.1; % mutation probability
> orelax = 1.4; % over relaxation factor for mating
> epsilon=0.07 % mutation range
> bestFitness=0;
> %chromPop % for debugging
> for g=1:100 % 100 generations
     % mutate each w/prob pm
     m = (rand(1,popSize)<pm); % mutation indices
     % mutate each chromoPop(i) within epsilon
     a = max(lower,chromPop-epsilon);
     b = min(upper,chromPop+epsilon);
     r = m.*(a+(b-a).*rand(1,popSize)); % new value
     chromPop = (1-m).*chromPop; %null out old value
     chromPop = chromPop+r; %replace by new value
     %chromPop % for debugging
     % mate each chromosome via a permutation of chromPop
     permPop = chromPop;
     for i=1:popSize-1
     k = i+floor((popSize+1-i)*rand); %random in remaining
     swap = permPop(i);
     permPop(i) = permPop(k);
     permPop(k)=swap;
   end
> %permPop % for debugging
>  newPop = chromPop +···
   orelax*(permPop-chromPop).*rand(1,popSize);
>  % stay within 0 and 1
>  newPop = max(lower,newPop);
>  newPop = min(upper,newPop);
> %newPop % for debugging
>  % evaluate newPop
>  fitnesses = kellyEval(popSize,newPop);
>  oldBest = bestFitness;
>  bestFitness = max(oldBest,max(fitnesses));
```

```
continued
>  if bestFitness > oldBest
      m = fitnesses>=bestFitness; % where is it
      solution = m.*newPop;
   end
> % roulette wheel select, make up the wheel
>  sum = 0;
>  for i=1:popSize
      sum = sum + fitnesses(i);
      bp(i) = sum;
   end
> %bp % for debugging
>  % now pick popSize times
>  for i=1:popSize
      U=rand; k=1;
      while( U>= bp(k))
         k=k+1;
      end; %k is selected
      chromPop(i) = newPop(k);
   end
> %chromPop % for debugging
> end
> bestFitness
> solution
```

5.6 Random Walks and Electrical Networks

Surprisingly, there are connections between random walks on a graph and electrical networks on the same graph. The standard reference for this connection is the very good book [DS84], on which most of this section is based.

Return to the gambler's ruin problem and recall that the recurrence equations for the probability of breaking the house h_x in a fair game are

$$h_x = \frac{1}{2}h_{x-1} + \frac{1}{2}h_{x+1},$$
$$h_0 = 0, \qquad h_a = 1,$$

where the gambler starts with x units of money and bets one unit on each play. In this, a is all the money in the game, the sum of the gambler's fortune and the house's fortune, and is constant.

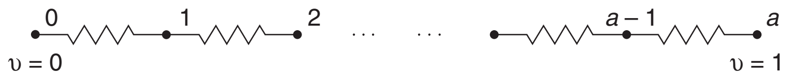


Fig. 5.7. One-dimensional electrical network.

Now consider an electrical circuit consisting of $a-1$ resistors of resistance R connected in series across a 1-volt battery; see Figure 5.7. Let $v(x)$ be the voltage at node x as shown. Then $v(0)=0$ and $v(a)=1$. By Kirchhoff's law, the sum of the current i into each node must be zero (since no current is stored in the node). And according to Ohm's law applied to the voltage v between two nodes, $v=iR$, so at node x we have

$$\frac{v(x-1)-v(x)}{R}+\frac{v(x+1)-v(x)}{R}=0.$$

Solving for $v(x)$ gives

$$v(x)=\frac{1}{2}v(x-1)+\frac{1}{2}v(x+1).$$

Thus h and v satisfy the same system of linear equations. Since the solution of a nonsingular linear system is unique, we must have $v(x)=h_x$, $x=0,1,\dots,a$, or the voltages in this simple circuit are the same as the probabilities of breaking the house!

Two-Dimensional Resistor Networks

Consider now a resistor network on integer-valued points, or nodes, in a region of the plane, for example as in Figure 5.8. An interior point of the network is linked by a resistor to four other nodes. For convenience we assume that all the resistors of the network have resistance $R=1$ ohm. The network is bounded by a set of nodes designated O having voltage 1 and another set designated Z at voltage 0. As above, let $v(x,y)$ be the voltage at node (x,y).

For an interior point (x,y), by Kirchhoff's and Ohm's laws, we can write

$$\frac{v(x+1,y)-v(x,y)}{R}+\frac{v(x,y+1)-v(x,y)}{R}+\frac{v(x-1,y)-v(x,y)}{R}+\frac{v(x,y-1)-v(x,y)}{R}=0,$$

since the sum of the currents into (x,y) is zero. Upon rearrangement this becomes

$$v(x,y)=\frac{1}{4}\left(v(x+1,y)+v(x-1,y)+v(x,y+1)+v(x,y-1)\right). \tag{5.17}$$

Of course, v is 1 on nodes O and 0 on nodes Z.

A function defined on a bounded lattice of nodes in a connected region of the plane is called *harmonic* if it satisfies (5.17) at each interior point of the lattice. So v is a harmonic function.

An important fact about harmonic functions is that since their value at every interior point is the average of their values at the neighboring points, the maximum and minimum of a harmonic function must occur on the boundary.

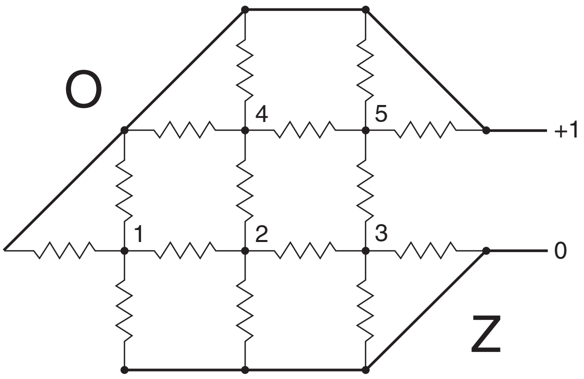


Fig. 5.8. Two-dimensional electrical network.

Now consider a random walk on the network. Let h_{xy} be the probability that starting from (x, y), the walk reaches a node of the set O before reaching a node of Z. Then $h_{xy} = 1$ if $(x, y) \in O$ and $h_{xy} = 0$ if $(x, y) \in Z$. For an interior point we have

$$h_{xy} = \frac{1}{4}h_{x+1,y} + \frac{1}{4}h_{x-1,y} + \frac{1}{4}h_{x,y+1} + \frac{1}{4}h_{x,y-1}, \tag{5.18}$$

since a step to the left, right, up, or down occurs with probability 1/4. Thus once again we see that $v(x, y) = h_{xy}$ for all points of the lattice. This shows that the voltages on the electrical network can be found by conducting random walks on the network!

As before, we may ask whether the walk could continue indefinitely or will end in the set $O \cup Z$ with certainty. Using the properties of harmonic functions it is easy to see that with probability one the walk will end. Let r_{xy} be the probability of gambler's ruin (in this case, that the walk will end up at the boundary component Z) starting from (x, y). It is a straightforward calculation to show that the sum $r_{xy}+h_{xy}$ is a harmonic function, since each is such a function individually. So $r_{xy}+h_{xy}$ must take its maximum and minimum values on the boundary of the region, that is, on either O or Z. But this sum is exactly 1 on every point of O and Z. Therefore $r_{xy}+h_{xy}=1$ for all (x,y) throughout the region!

Following these principles, a simulation of 100,000 random walks on the network of Figure 5.8 produced the results shown in Figure 5.9.

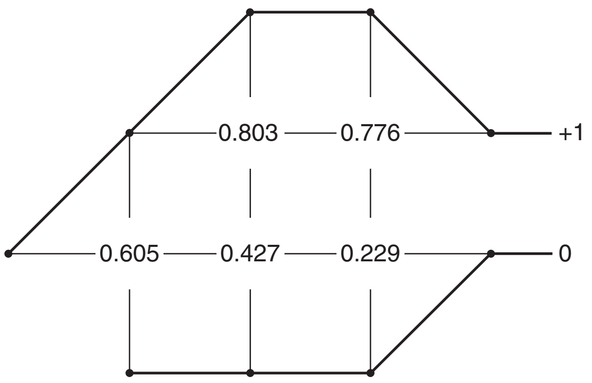


Fig. 5.9. Solution of the 2D electrical network.

Although a random walk may be used for solving such an electrical network, it is not the most efficient method for the task. On a very large region with many nodes it will take a great deal of computational time for the walk to reach a point of the boundary. Nevertheless, it can be an effective method under some circumstances, for example when the solution is required at only a small number of nodes. It should also be noted that this Monte Carlo algorithm, like nearly all others, can be easily and effectively parallelized. Furthermore, it is very interesting to think about the connection and helps us to understand both random walks on grids and electrical circuits.

The Dirichlet Problem

The *Dirichlet problem* is that of finding a function harmonic throughout a bounded, connected (continuum) region and having prescribed values on the boundary of the region. We have defined the term harmonic for a function restricted to a lattice of points in the plane, but the term can also be defined for functions on the continuum as well, that is, for all real-valued points (x,y) in the region. In fact, this was done in the section on diffusion as solutions to (5.2).

If a numerical solution is desired to the Dirichlet problem in the continuum, one technique of solution, that of *finite elements*, discretizes the region, forming a lattice, and solves for a solution there. The Monte Carlo method could be used for solving this discretized problem. However, as noted above, there are more effective numerical methods for solving the Dirichlet problem.

5.6.1 Markov Chain Solution for Voltages

Referring to Figure 5.8, consider the Markov chain defined over the nodes of the network $\Omega = \{O, Z, 1, \dots, 5\}$. Since we are interested in the dynamics of reaching states O and Z, we regard them as *absorbing states*, that is, the probability of a transition out of one of these states is zero. With this consideration, the probability transition matrix is

$$P = \begin{array}{c} \\ \text{O} \\ \text{Z} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{cc|ccccc} \text{O} & \text{Z} & 1 & 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{cc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline \frac{1}{2} & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} \\ \frac{1}{2} & 0 & 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 \end{array}\right] \end{array}. \tag{5.19}$$

As before, let h_{xy} be the probability of reaching state O before state Z starting from node (x, y), and recall (5.18). However, it is notationally simpler to use the labels from Figure 5.8 instead; node $(1, 1)$ is 1, $(1, 2)$ is 2, and so on. With relabeling, the unknowns are $h_1, \dots, h_5$. Completing the set of equations are the conditions on the boundaries O and Z,

$$h_O = 1, \qquad h_Z = 0. \tag{5.20}$$

The relabeled interior equations (5.18) become

$$\begin{aligned}
\frac{1}{4}h_O + \frac{1}{4}h_O + \frac{1}{4}h_Z + \frac{1}{4}h_2 &= h_1, \\
\frac{1}{4}h_Z + \frac{1}{4}h_1 + \frac{1}{4}h_3 + \frac{1}{4}h_4 &= h_2, \\
\frac{1}{4}h_Z + \frac{1}{4}h_Z + \frac{1}{4}h_2 + \frac{1}{4}h_5 &= h_3, \\
\frac{1}{4}h_O + \frac{1}{4}h_O + \frac{1}{4}h_2 + \frac{1}{4}h_5 &= h_4, \\
\frac{1}{4}h_O + \frac{1}{4}h_O + \frac{1}{4}h_3 + \frac{1}{4}h_4 &= h_5.
\end{aligned}$$

Add to this system the identities $h_O = h_O$ and $h_Z = h_Z$, and we may write the system in matrix form in terms of P,

$$P\mathbf{h} = \mathbf{h}. \tag{5.21}$$

Here we are writing the vector on the right and taking $\mathbf{h}$ to be the column vector $[\,h_O \quad h_Z \quad h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5\,]^T$. We see that $\mathbf{h}$ is a fixed point of P (and a right eigenvector of P with corresponding eigenvalue 1). Note the similarity of this to the equation for an invariant distribution, equation (3.5). Therefore

$$P^k\mathbf{h} = \mathbf{h}$$

for all $k = 1, 2, \ldots$.

Now partition the states into the absorbing states versus the nonabsorbing states. The partition induced on $\mathbf{h}$ is

$$\mathbf{h} = \left[\begin{array}{cc|ccccc} h_O & h_Z & h_1 & h_2 & h_3 & h_4 & h_5 \end{array}\right]^T = [\,\mathbf{h_B} \quad \mathbf{h_D}\,]^T,$$

where $\mathbf{h_D} = [\,h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5\,]$ corresponds to the interior of the domain and $\mathbf{h_B} = [\,h_O \quad h_Z\,]$ to the boundary. Likewise, partition the transition matrix P as shown in (5.19),

$$P = \begin{bmatrix} I_2 & 0 \\ R & \hat{P} \end{bmatrix}. \tag{5.22}$$

Here I_2 is the 2×2 identity matrix, 0 is a 2×5 zero matrix, R is a 5×2 nonabsorbing to absorbing transition matrix, and $\hat{P}$ is the 5×5 transition matrix for the nonabsorbing states,

$$\hat{P} = \begin{bmatrix} 0 & \frac{1}{4} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} \\ 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 \end{bmatrix}.$$

We are now ready to solve for $\mathbf{h_D}$. Note that

$$P^2 = \begin{bmatrix} I_2 & 0 \\ R + \hat{P}R & \hat{P}^2 \end{bmatrix}$$

and

$$P^3 = \begin{bmatrix} I_2 & 0 \\ R + \hat{P}R + \hat{P}^2R & \hat{P}^3 \end{bmatrix},$$

and so on for the powers of P. Let

$$B = R + \hat{P}R + \hat{P}^2R + \hat{P}^3R + \cdots = (I + \hat{P} + \hat{P}^2 + \ldots)R.$$

Just as with ordinary geometric series for numbers, the geometric series of matrices in this equation sums to $(I - \hat{P})^{-1}$, where I is the identity matrix of the same size as $\hat{P}$ (it is interesting to think about why this is the case). Hence

$$B = (I - \hat{P})^{-1}R. \tag{5.23}$$

Further, since $\hat{P}$ is a nonnegative matrix with row sums less than 1, from the Perron–Frobenius theorem we can say that $\hat{P}^k \to 0$ as $k \to \infty$. So in the limit we can write

$$\begin{bmatrix} \mathbf{h_B} \\ \mathbf{h_D} \end{bmatrix} = \mathbf{h} = \lim_{k\to\infty} P^k\mathbf{h} = \begin{bmatrix} I_2 & 0 \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathbf{h_B} \\ \mathbf{h_D} \end{bmatrix}.$$

From this we get

$$\mathbf{h_D} = B\mathbf{h_B} = (I - \hat{P})^{-1}R\mathbf{h_B}.$$

Let $N = (I - \hat{P})^{-1}$. This matrix is called the *fundamental matrix*; and from Matlab we calculate

$$B = NR = \begin{bmatrix} 0.6067 & 0.3933 \\ 0.4270 & 0.5730 \\ 0.3006 & 0.6994 \\ 0.8006 & 0.1994 \\ 0.7753 & 0.2247 \end{bmatrix}.$$

By (5.20), $h_O = 1$ and $h_Z = 0$, so

$$\mathbf{h_D} = \begin{bmatrix} 0.6067 & 0.3933 \\ 0.4270 & 0.5730 \\ 0.3006 & 0.6994 \\ 0.8006 & 0.1994 \\ 0.7753 & 0.2247 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.6067 \\ 0.4270 \\ 0.3006 \\ 0.8006 \\ 0.7753 \end{bmatrix}.$$

Thus the solutions to the Dirichlet problem for this case are 0.6067 at $(1,1)$, 0.4270 at $(1,2)$, 0.3006 at $(1,3)$, 0.8006 at $(2,2)$, and 0.7753 at $(2,3)$. That is, these values represent both the voltages at these locations (given a voltage of 0 at the boundary Z and a voltage of 1 at the boundary O) and the probability that a random walk started at the given location will arrive at the boundary O before arriving at the boundary Z. Compare these values with those obtained from simulating many random walks and given in Figure 5.9.

```
Matlab
> Ph = [0 1/4 0 0 0; 1/4 0 1/4 1/4 0;...
   0 1/4 0 0 1/4; 0 1/4 0 0 1/4; 0 0 1/4 1/4 0]
> N = (eye(5)-Ph)^-1
> R = [1/2 1/4; 0 1/4; 0 1/2; 1/2 0; 1/2 0]
> N*R
```

5.6.2 The Fundamental Matrix and Expected Hitting Times

Label the nonabsorbing states of a Markov chain as $1, 2, \dots, n$ and let E_k be the expected number of iterations for the chain to reach an absorbing state if the chain is started in state k. Let A denote the set of absorbing states and p_{kA} the probability of transition from k to an absorbing state in one iteration; thus

$$p_{kA} = \sum_{a \in A} p_{ka}.$$

By decomposing the chain on one iteration we get the following equation for E_k:

$$E_k = p_{kA} + p_{k1}(1 + E_1) + p_{k2}(1 + E_2) + \cdots + p_{kn}(1 + E_n). \quad (5.24)$$

This is because with probability p_{k1} this transition will be to state 1, and if so, the number of iterations to be absorbed is 1 for this transition plus the expected number needed from state 1, and so on for the other possible transitions. Letting $\mathbf{E}$ be the column vector of the E_k's, then (5.24) may be rewritten in matrix form as

$$\begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} = \begin{bmatrix} p_{1A} + p_{11} + \ldots + p_{1n} \\ p_{2A} + p_{21} + \ldots + p_{2n} \\ \cdots \\ p_{nA} + p_{n1} + \ldots + p_{nn} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix} \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}.$$

Note that the sums $p_{jA} + p_{j1} + p_{j2} + \cdots + p_{jn}$ equal 1 for each j. As before, denoting the transition matrix among the *transient states* by $\hat{P}$, this becomes

$$(I - \hat{P})\mathbf{E} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix},$$

or

$$\mathbf{E} = (I - \hat{P})^{-1}\mathbb{1} = N\mathbb{1}, \quad (5.25)$$

where $\mathbb{1}$ is the column matrix of 1's. This gives a simple expression for the expected hitting times from any starting state to the set of absorbing states. Notice that equation (5.25) depends only on the transitions between nonabsorbing (or *transient*) states. This makes perfect sense, since the higher the probability for transitions between transient states, the longer one expects it to take to reach the absorbing states.

Equation (5.25) can also be used to compute an *expected hitting time* to an arbitrary collection of states for a general Markov chain (either with or without absorbing states). Say that some subset of states $G \subset \Omega$ is singled out as being important (such as the set of global minima for an optimization problem, for example). Modify the transition matrix P by making each $g \in G$ into an absorbing state. Using the above technique, one can theoretically calculate the expected hitting time to reach the subset G.

Application to Genetic Algorithms

The matrix partition (5.19) between absorbing and nonabsorbing states has a very important application to Monte Carlo optimization. We illustrate this for genetic algorithms. Regard the goal states of an optimization as the absorbing states. Then the nongoal states are the nonabsorbing or transition states. As in (5.19), let P be the transition matrix of the GA and partition it so that $\hat{P}$ is the transition matrix among the nongoal states.

Let the invariant vector π be likewise partitioned $\pi = [\,\alpha \quad \beta\,]$. From $\pi P = \pi$ we get $\beta \hat{P} = \beta$.

Invoking the Perron–Frobenius theorem, let λ be the principal eigenvector of $\hat{P}$ and let ω be the corresponding positive (left) eigenvector normalized to sum to 1. Since the row sums of $\hat{P}$ are less than or equal to 1, and some are strictly less than 1, then by the Perron–Frobenius theorem, $\lambda < 1$. We then have

$$\omega \hat{P} = \lambda \omega.$$

In terms of probability mass, this says that starting with 1 unit of probability mass, after one iteration of the chain only λ fraction of it remains. Therefore $1 - \lambda$ is the rate at which probability mass leaves the nongoal states, that is, the probability that the chain will find a goal state per iteration once startup transients have dissipated away. See Problem 18 at the end of this section for more information.

General Resistor Networks and Reversible Markov Chains

Let G be a connected graph, that is, a finite set of vertices x, y, …, z connected by edges such that it is possible to go between any two vertices traveling along the edges. The graph G becomes a resistor network by installing resistors along its edges. The vertices x of the graph become the nodes of the resistor network, at which places the voltage v_x may be measured.

If xy is an edge, let R_{xy} denote the resistance connecting nodes x and y. Of course $R_{yx} = R_{xy}$. The reciprocal of resistance, $C_{xy} = 1/R_{xy}$, is defined as the *conductance* between x and y. According to Ohm's law, the current i_{xy} from x to y is calculated as $(v_y - v_x)C_{xy}$. Let $C_x = \sum_y C_{xy}$ be the sum of the conductances out of x leading to all possible neighboring nodes y.

A Markov chain may be associated with this network as follows. The states of the chain are the nodes of the network, the nonzero transitions from x are exactly the neighbors of x in the network, and the transition probabilities are defined by

$$p_{xy} = \frac{C_{xy}}{C_x}, \tag{5.26}$$

i.e., that edge's fraction of the node's conductance.

The resulting chain is irreducible (but may not be aperiodic). Nevertheless, just as in Chapter 3, it has a unique invariant distribution π, a probability vector satisfying $\pi P = \pi$. In fact the components of the invariant vector are given by $\pi_x = C_x/C$, where C is defined as $C = \sum_x C_x$. A characterizing property of electrical network chains is that for all x and y,

$$\pi_x p_{xy} = \pi_y p_{yx}.$$

This is the same *detailed balance* that we have already seen in Section 3.2.

Detailed balance follows because

$$\pi_x p_{xy} = \frac{C_x}{C}\frac{C_{xy}}{C_x} = \frac{C_y}{C}\frac{C_{xy}}{C_y} = \pi_y p_{yx}.$$

Alternatively, the chain is said to be *reversible.* The reason is that when the chain reaches invariance, a path from x_1 to x_2 to … to x_n moving forward in time has the same probability of occurring as the reverse path. For example, for a path from x to y to z, by detailed balance is given by

$$\pi_x p_{xy} p_{yz} = \pi_y p_{yx} p_{yz} = p_{yx} \pi_y p_{yz} = p_{yx} \pi_z p_{zy} = \pi_z p_{zy} p_{yx},$$

and so we see that the forward and reverse paths have the same probability of occurring.

The converse is also true: a Markov chain satisfying detailed balance corresponds to an electrical (resistor) network just as above. One only has to put $C_{xy} = \pi_x p_{xy}$ for all terms of P. Note that if P has a nonzero diagonal term $p_{xx} \neq 0$, then the circuit will have a self-connected node at x of conductance $\pi_x p_{xx}$. (A network is not prohibited from having such a self-loop, but no current will flow through it.)

Now consider a random walk on G. The walker at node x selects among its neighbors y according to roulette wheel selection with probabilities p_{xy}. The electrical interpretation is that of an electron moving randomly from node to node under the influence of the conductances. The random walk is therefore just the Markov chain state random variable $X(t)$, and the component π_x of the invariant distribution is the asymptotic frequency with which the walker visits state x.

Next select two nodes of the network a and b and connect a 1-volt battery across these nodes with $v_a = 1$ and $v_b = 0$. The corresponding modification of the Markov chain is to make the states a and b absorbing. As before, let h_x denote the probability that the random walk starting at x hits node a before node b. Then, as before, h_x equals the voltage v_x of the network at x.

To see this, by invoking Kirchhoff's and Ohm's laws at node x we have

$$0 = \sum_y (v_y - v_x) C_{xy} = \sum_y v_y C_{xy} - v_x \sum_y C_{xy}.$$

Hence, letting $C_x = \sum_y C_{xy}$,

$$v_x = \sum_y \frac{C_{xy}}{C_x} v_y = \sum_y p_{xy} v_y.$$

This shows that v_x is the average of its neighboring values and so is a harmonic function for all $x \neq a, b$. Likewise, h_x is harmonic for all $x \neq a, b$. Further, $v_a = h_a = 1$ and $v_b = h_b = 0$. It follows that v and h must agree for all x.

Notes

The remarkable fact that diffusion can be derived from a simple random walk on the line is yet another consequence of the central limit theorem. There is an alternative derivation of the diffusion equation via Stirling's approximation to the factorial, namely

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n .$$

The Wiener process is the starting point for an entire calculus defined on functions. Wiener measure assigns a "size" to sets of functions just as ordinary calculus assigns a "size" to sets of points such as intervals (a, b). One can then integrate with respect to Wiener measure. On the other side of calculus is the derivative. Equation (5.4) for the moment-by-moment description of the stock price is the derivative of Brownian motion. This is an example of a *stochastic differential equation*. Just as ordinary differential equations have solutions, so it is with their stochastic counterparts. The solution of (5.4) is the famous Black–Scholes option-pricing equation. With notation as in Section 5.3, for a call option, the price C is given by

$$C = SF(d_1) - Xe^{-rT}F(d_2),$$

where $F(\cdot)$ is the $N(0,1)$ cumulative distribution function, Table 2.2, and d_1 and d_2 are

$$d_1 = \frac{\log(\frac{S}{X}) + (r + \sigma^2/2)T}{\sigma\sqrt{T}},$$
$$d_2 = \frac{\log(\frac{S}{X}) + (r - \sigma^2/2)T}{\sigma\sqrt{T}}.$$

The Kelly criterion has been advocated as the basis of an investment system because it optimizes the long-term growth rate of the investment. However, some financial theorists, most notably Paul Samuelson, are strongly opposed to this. Their fears are based on the fact that as a random walk, investments can, and with probability 1 will, shrink to fractions of the starting value. It is quite unfortunate if one's retirement occurs during such a time. Instead, most advisors use the system developed by Harry Markowitz known as the capital asset pricing model. The strategy here is to invest in a diversified portfolio whose investments are chosen in relation to the risk that the investor is willing to bear. For a delightful and fascinating account of the history of the Kelly criterion see "Fortunes Formula" [Pou05].

Probability has always been associated with gambling, and as noted in the text, had its roots in gambling. It is no surprise that much has been written about gambling systems. One of the most influential treatises in this area is "How to Gamble If You Must," by Lester E. Dubbins and Leonard J. Savage.

Problems: Chapter 5

1. (4) (Leads in walk) For a random walk starting at the origin, by a "leads in walk" we mean to say "heads" are leading if the present position is positive and "tails" are leading if the present position is negative. As the experiment, keep a running count of whether heads or tails is leading for each step of a total walk of 1000 steps. Histogram the fraction of the walk in which heads is leading for some number of trials of this experiment. Comment on the shape of the histogram. Is it Gaussian? Repeat again with 100 trials of 10,000 steps each.

2. (4) (Diffusion through a crack) Simulate an unbiased random walk ($\Pr(H) = \Pr(T) = 0.5$) with some obstacles. (a) Put a reflecting barrier at $x = -4$, i.e., the walk may land at -4 but may not stay there, and the next step must be to -3. Plot a sample path and a histogram. (b) Next add a partial block at $x = +6$, i.e., at $x = 6$ there is only a 0.25 chance of going to 7 and a 0.75 chance of going to 5 (same for 7 to 6). Plot a histogram. These are results that are hard to get in any way other than by simulation.

3. (5) For a random walk on the set $\Omega = \{0, 1, 2, \ldots, a\}$, with probability that $p = 0.51$ of taking a step to the right, let h_x denote the probability that the walk will reach a before reaching 0. (a) Calculate h_x by Markov chain methods. (b) Use simulation to estimate h_x.

4. (6) (Wiener process/Brownian motion) Simulate a random walk in the plane as follows. Start at the origin. From any given position, choose a direction from 0 to 360 degrees equally likely, that is, uniformly on $[0, 360)$. Next choose a step size according to a Gaussian (normal) distribution with mean 0 and variance σ^2. Advance to that position and continue. Take σ^2 to be, variously, 0.5, 1, 2, the idea being to see what effect variance has on the walk. Take the number of steps to be, variously, 20, 400, 1600, again to gauge the effect. Show a typical walk (i.e., a sample path) for one of the 1600 (or more) steps. Also, instead of histogramming, show a density plot, that is, for several walkers, plot a point in the plane where the walker stops.

5. (3) (Black–Scholes) A difference equation for the future price S of a stock is
$$\frac{\Delta S}{S} = \mu \Delta t + \sigma \Delta W.$$
In this, μ is the growth rate of the stock, σ is its *volatility*, and ΔW is a normally distributed random variable with mean 0 and variance Δt. (Alternatively, $\Delta W = \sqrt{\Delta t} N$ where $N \sim N(0, 1)$.) Taking $\mu = 0.1$ per year, $\sigma = 0.3$ per square root year, and the starting price $S(0) = 100$, simulate the random walk with 1-day increments, $\Delta t = 1$, to find the distribution of the price of the stock $S(T)$ in $T = 90$ days. (Don't forget to divide the

growth rate by 365 and volatility by square root of 365 to get the per day values.)

6. (4) (Black–Scholes) With the setup as in Problem 5, take various values of σ and of T to gauge the effect these parameters have on the distribution. Comment.

7. (5) (Black–Scholes) With the setup as in Problem 5, and using a risk-free rate of 3.1%, calculate the value of call options for various strike prices from \$90 to \$115. See Section 5.3. Graph the results with respect to each parameter: volatility, expiration, and strike price.

8. (5) (Black–Scholes) Do the same as the Problem 7 except for put options. The only difference is that for puts, the payoff is $\max(X - S_T, 0)$. That is to say, if the ending price S_T is less than the strike price X, then the option is in-the-money by the value of their difference.

9. (6) (Diffusion in the plane) (a) Starting at the origin in the plane, carry out a random walk on the lattice of integer coordinate points over the square $[-20, 20] \times [-20, 20]$. Assume that there is a square hole whose boundary is $(13, 7)$, $(14, 7)$, $(14, 8)$, and $(13, 8)$. Walks that reach this hole are absorbed, along with any walk which reaches the boundary. Show the distribution of final positions for walks of various steps and especially the fraction that enters the hole. (b) Same thing but this time assume a reflecting obstacle at $y = 6$ and $-12 \le x \le 12$.

10. (5) (Levy flight) For the Cauchy density $f(x) = \frac{(2/\pi)}{1+x^2}$, $x \ge 0$, there is a high probability of selecting a small value and a low, but not 0, probability of selecting a large value. Perform a random walk in the plane, starting at the origin, in which the direction of the next step is uniform at random from 0 to 360 degrees and the step size is from f. Such a random walk is called a *Levy flight*. Draw samples from f by inverting the cdf. In this regard, note that for $x \ge 0$, the integral of f is

$$F(x) = \int_{-\infty}^{x} \frac{(2/\pi)}{1+t^2} dt = \int_{0}^{x} \frac{(2/\pi)}{1+t^2} dt = \frac{2}{\pi} \tan^{-1} x.$$

Show a typical walk (i.e., a sample path) for some convenient number of steps (try a few hundred, experiment some). Look for self-similarity in the plots. Also, instead of histogramming, show a density plot, that is, for several walkers, plot a point in the plane where the walker stops. (It is thought that seabirds use this random walk in searching for schools of fish.)

11. (5) (Levy flight) Discover the empirical transport properties of Levy flight (see Problem 10), that is, how far from the origin does a random walker

get, on average, as a function of the number of steps n. Do this by simulating a few hundred flights for various values of n and plotting the average distance displaced versus n.

12. (5) (Gambling system.) Suppose you use the system "double up" to win \$1 at Las Vegas roulette. The wheel has 18 red, 18 black and 2 green sectors, and there is an equal chance that any one sector is selected. Thus to start, bet \$1 on "red," say. If you win (with probability 18/38), stop. Otherwise double your previous bet, e.g., \$2 on the second time, until you do win. Simulate this to find out (1) the maximum number of bets before winning the \$1, (2) if the house has a betting limit of \$1,024, how often do you lose everything? (Expected number of turns to win: $p + 2qp + 3q^2p + \cdots = p/(1-q)^2 = 1/p$, where $p = 18/38$ the probability of winning.)

13. (5) The same as in Problem 12, but now suppose the probability of winning is $p = 1/2$. Assume that the house has unlimited funds but the player can bet only up to \$1024 (i.e., the gambler starts with \$2047).

14. (5) Simulate (a reasonable approximation of) the game of blackjack (or 21) as follows. A game begins with the player receiving two cards. With aces counting 11, jacks and queens and kings (known as "face" cards) counting 10, and all other cards counting their index value, the value of the player's hand is the sum of the count of his cards. The object of the game is that the value should be closer to 21, without exceeding 21, than the dealer's hand. If the value is 21, the player wins immediately. Otherwise, the player may elect to be "hit," i.e., receive, one at a time, additional cards. If at any time the value exceeds 21, the player loses. When satisfied, the player "stands" and receives no more cards. Next the dealer receives two cards. The house rule is that the dealer is to stand on 17 or better and draw on 16 or less. If the dealer breaks 21, the player wins; the dealer wins if his value exceeds or equals the player's; otherwise, the player wins.
    Simulate several such games using a single deck or "shoe" of 52 cards. Does the player or dealer come out ahead in the long run? Card-counting players of this game attempt to count the number of "10's," i.e., 10's or face cards, as the shoe works down, and alter the size of their bets accordingly by betting bigger when the deck is 10's "rich."

15. (5) (Self-avoiding walk) Conduct a self-avoiding random walk on a square lattice. (a) Estimate c_n for $n = 4, 5, 6$, and 7. (b) Also estimate the expected squared distance from the origin of the endpoint of the walk for these values of n. (c) Make a plot of average of the displacement squared vs c_n.

16. (6) (Noisy "Wire," Kelly criterion) In earlier times a gambler could get advance notice of the winner of a horse race from an informant at the track and use that information to place bets off track. But the wire is noisy and the gambler gets the information wrong 20% of the time. (That is, the information "bet on X to win" loses 20% of the time.) The gambler starts with a stake of S dollars. (a) By simulation, what fraction of his fortune should the gambler risk each time, assuming that there will be a large number of such bets, to maximize the expected growth rate of his fortune? (b) Plot one of the possible fortune histories. Does it ever reach one-half the gambler's original stake S? How about one-third? (c) Why is it better to follow Kelly's criterion than to try to maximize the expected winnings? Test your answer using simulation with different betting strategies. Which one performs best and why?

17. (6) (Random walk on grid) Simulate a random walk on the points with integer coordinates bounded by the triangle x-axis, y-axis and the line joining $(0, 9)$ with $(12, 0)$. Let those points nearest the hypotenuse count as "1" and those on the x-axis and y-axis as "0." (Specifically, in addition to the other points on the two axes, the points $(0, 9)$ and $(12, 0)$ are to be 0's; the 1's are $(1, 8)$, $(2, 7)$, $(3, 7)$, $(4, 6)$, $(5, 5)$, $(6, 4)$, $(7, 4)$, $(8, 3)$, $(9, 2)$, $(10, 1)$, and $(11, 1)$.) To do: (a) for each point (i, j) of the interior of the triangle, walk until you reach a "0" or a "1", then stop and note the outcome 0 or 1; estimate the expected outcome for (i, j). (b) estimate the expected hitting time from each point.

18. (6) In Problem 1 of the optimization chapter, Chapter 4, the transition matrix for a simulated annealing solution of the 2-bit Sandia Mountain problem is given. Instead of lowering the temperature, assume that it is fixed for this problem; then the material of Section 5.6 applies.
Decompose this matrix, in the same way as that of equation (5.19). The goal state $x = 0$ acts as the absorbing state, and $\hat{P}$ is the transition matrix among the nongoal states. For some fixed temperature, use simulated annealing to find the goal starting equally likely in a non-goal state. Use your simulation to estimate (a) the rate per iteration of finding the goal and (b) the expected number of iterations to find the goal. Compare these estimates of λ and $\mathbf{E}$ with exact values, obtained as in Section 5.6.2.

19. (8) (Kelly criterion with correlation) In the text it was assumed that the two option trades were independent. But this is often not true. The trend of the market in general affects a majority of the stocks in the same way. In this case we say that the two trades are *correlated.* The correlation between two random variables X and Y is measured by the *correlation coefficient* ρ,

$$\rho = \frac{\operatorname{cov}(X, Y)}{\sqrt{\operatorname{var}(X)\operatorname{var}(Y)}}. \tag{5.27}$$

Recall covariance from (1.24). If X and Y are Bernoulli trials and $f(1,1)$ is the joint probability of success, then (5.27) becomes

$$\rho = \frac{f(1,1) - p_X p_Y}{\sqrt{p_X q_X p_Y q_Y}}, \tag{5.28}$$

where p_X and p_Y are the marginal success probabilities for X and Y respectively and $q_X = 1 - p_X$ and $q_Y = 1 - p_Y$ as usual.
For this problem, rework the two option trades example making the assumption that they jointly vary with $\rho = 0.8$. Assume that the trades only win or lose, i.e., there is no catastrophic loss category. Letting

$$\eta = \rho\sqrt{p_X q_X p_Y q_Y}, \tag{5.29}$$

the joint win(1) or loss(0) probabilities for (X, Y) become

$$\begin{aligned} f(1,1) &= p_X p_Y + \eta, \quad & f(0,0) &= q_X q_Y + \eta, \\ f(1,0) &= p_X q_Y - \eta, \quad & f(0,1) &= q_X p_Y - \eta. \end{aligned}$$

Now the success probabilities are the marginal successes.

A

Generating Uniform Random Numbers

A.1 Multiple Stored Value Random Number Generation

As we saw above in Section 1.3.3, the period of the linear congruential RNG is necessarily at most m, the value of the modulus. This is forced by the use of only one stored value, or seed; when the seed equals any value that it had on a previous iteration, then the sequence of returned pseudorandom numbers will repeat.

However by using two or more stored values, the period can greatly exceed m. In this section we look at some of the possibilities.

A.1.1 Fibonacci Generators

We start by incorporating two stored values, $seed1$ and $seed2$. Additionally there are also three parameters a_1, a_2, and c, and a modulus m. Initially the generator is doubly seeded; let $x_1 = seed1$ and $x_0 = seed2$. The next number generated is

$$x_{n+1} = (a_1 x_n + a_2 x_{n-1} + c) \bmod m.$$

In this way a sequence $x_2, x_3, \ldots, x_n, \ldots$ of pseudorandom numbers is generated. Often c is taken to be 0.

These types of RNGs are referred to as *Fibonacci generators* in reference to the famous Fibonacci sequence $x_n = x_{n-1} + x_{n-2}$ starting with $x_0 = 0$ and $x_1 = 1$, which was proposed centuries ago by Fibonacci to model the population growth of rabbits.

More generally, one can create a table of stored values and allow more flexibility in their choice. For example, good results are reported for the generator

$$x_n = (x_{n-24} + x_{n-55}) \bmod m, \qquad n \geq 55,$$

where the multipliers are both 1. This generator requires seeding a table with the values x_0, x_1, …, x_{54}.

Another example is the Fibonacci generator proposed by Marsaglia given in Figure A.1. In this code, the syntax `a % b` means a mod b.

Marsaglia Random Number Generator

```
double rndm(long ss)
//call with $ss \neq 0$ to seed, else with $ss = 0$
  static int nBits = 32; // set to # bits used for integers
  static long m1 = $2^{\mathtt{nBits}-2} + 2^{\mathtt{nBits}-2} - 1$;
  static long m[17]; // 17 components must be initialized
  static long i, j;
  long seed, k, k0, k1, j0, j1, m2 = $2^{\mathtt{nBits}/2}$;
  if( ss $\neq$ 0 ) {
    if( ss < 0 ) ss = -ss;
    seed = ss % m1;
    if( seed % 2 == 0 ) seed = seed-1;
    k0 = 9069 % m2;    k1 = 9069/m2;
    j0 = seed % m2;   j1 = seed/m2;
    loop p = $0, 1, \ldots, 16$
      seed = j0*k0;
      j1 = ((seed/m2)+j0*k1+j1*k0)% (m2/2);
      j0 = seed % m2;
      m[p] = j0+m2*j1;
    end loop
    i = 4; j=16;
  }
  k=m[i]-m[j];
  if( k<0 ) k = k+m1;
  m[j] = k;
  i = i-1; if( i<0) i=16;
  j = j-1; if( j<0) i=16;
  return (double)k/m1;
```

Fig. A.1. Marsaglia's generator

The key line is the sixth from the bottom, which shows that the generator has the Fibonacci form,

next value = (some previous value) – (some other previous value)

and the new value enters the table of previous values.

Fibonacci generators are harder to analyze than linear congruential generators. In the Marsaglia generator the output numbers are modulo $m1$. In turn, $m1$ depends on the number of bits entered into line 3.

With nBits = 32, the number of integers available to the generator is $m1 = 2^{31} - 1 = 2{,}147{,}483{,}647$, counting 0 but not $m1$ itself. The period is far greater than this, but how big is it?

To see the operation of the generator, imagine nBits=3, so that $m1 = 3$ and the available integers are $0, 1, 2$. Further, let the length of the m array be taken to be 3, initialize it to $1, 2, 0$, and let i and j be initialized to $i = 1$ and $j = 0$. These initial variables have maximal period before repeating of $3^3 \times 3 - 3 = 78$. (Note that i and j are not independent but cycle in lock step.) Hence it is possible to inspect the complete sequence for this choice of nBits.

The first three return values (without dividing by $m1$), manually computed, are (in the following let dec(i) mean subtract 1 from i and a % b mean a mod b as above):

currently $m = 1, 2, 0, \quad i = 1,\ j = 0$
  $k = m_i - m_j = 1$, return 1
  update: $m_j = k = 1,\ \text{dec}(i)\ \%\ m1 = 0,\ \text{dec}(j)\ \%\ m1 = 2$
currently $m = 1, 2, 0 \quad i = 0,\ j = 2$
  $k = m_i - m_j = 1$, return 1
  update: $m_j = k = 1,\ \text{dec}(i)\ \%\ m1 = 2,\ \text{dec}(j)\ \%\ m1 = 1$
currently $m = 1, 2, 1 \quad i = 2,\ j = 1$
  $k = m_i - m_j = -1\ \%\ 3 = 2$, return 2
  update: $m_j = k = 2,\ \text{dec}(i)\ \%\ m1 = 1,\ \text{dec}(j)\ \%\ m1 = 0$

Continuing the computation yields the full length-78 sequence:

```
1 1 2 1 0 1 0 0 2 2 2 0 1 2 2 1 2 0 2 0 0
1 1 1 0 2 1 1 2 1 0 1 0 0 2 2 2 0 1 2 2 1
2 0 2 0 0 1 1 1 0 2 1 1 2 1 0 1 0 0 2 2 2.
0 1 2 2 1 2 0 2 0 0 1 1 1 0 2
```

However, inspecting this output shows the numbers cycle with a period of 26; this is easiest to see starting with the sequence $2, 2, 2$ and continuing until this sequence arises again.

A.1.2 Finite Field RNG

(This section uses some more advanced material; in particular, it is based on the theory of finite fields.)

Some Fibonacci RNGs can be analyzed with the help of the theory of finite fields. We illustrate the method to construct generators that use two stored values and whose period is $m^2 - 1$. Consider an example, kept small for illustration purposes. Using a modulus of $m = 5$, a period-24 sequence is possible (but not a period-25 sequence).[1]

As an example, we construct a pseudorandom sequence of period $24 = 5^2 - 1$ with digits from $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$. This construction can be generalized to construct sequences of length $m^j - 1$ for any odd prime m and $j \geq 1$.

Our construction takes place in $G_{25} = \{a_0 + a_1 x \,:\, a_0, a_1 \in \mathbb{Z}_5, x^2 = 3\}$, the field of polynomials over $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ modulo $x^2 + 2$. Similar to the use of the the term *modulo* above (as in $8 \equiv 3 \bmod 3$), here modulo x^2+2 means that two polynomials are counted as the same if their difference is divisible by $x^2 + 2$. Also notice that the coefficients come from $\mathbb{Z}_5$, the equivalence class of integers modulo 5. So $8x - 3$ is the same polynomial as $3x + 2$ because $3 = 8 \bmod 5$ and $2 = -3 \bmod 5$. As another example,

$$2x^5 + 2x^4 + 4x^3 + x^2 + 3 \equiv 3x + 4 \bmod x^2 + 2,$$

since

$$2x^5 + 2x^4 + 4x^3 + x^2 + 3 = 3x + 4 + (x^2 + 1)(2x^3 - 3x^2 + 2x + 4).$$

Note that the polynomial $x^2 + 2$ cannot be written as the product of two polynomials of smaller degree with coefficients in $\mathbb{Z}_5$. One says that $x^2 + 2$ cannot be factored over $\mathbb{Z}_5$. So, in the polynomials over $\mathbb{Z}_5$ modulo $x^2 + 2$, we may identify x^2 with -2 or alternatively, with 3.

In G_{25} we want to find special values of a_0 and a_1 such that $\beta = a_0 + a_1 x$ generates the nonzero elements of G_{25} as its powers, i.e., we want β to be a primitive element of G_{25}. Try $\beta = 2 + 3x$. Before testing it, note that with this choice, β is a root of the polynomial $z^2 - 4z - 3$, since

$$\begin{aligned}\beta^2 - 4\beta - 3 &= (2 + 3x)^2 - 4(2 + 3x) - 3 \\ &= (4 + 12x + 9x^2) - (8 + 12x) - 3 \\ &= 3 + 4x^2 = 3 + 4 \times 3 = 0.\end{aligned}$$

Therefore, wherever it occurs, β^2 can be replaced by $4\beta + 3$. Furthermore, since $2\beta = 4+6x = 4+x$, x can be replaced by $x = 2\beta-4 = 2\beta+1$. Then computing the powers of β, we obtain

[1] **Exercise:** Show that if m is prime, then for any integers $a \neq 1$ and c, $x \mapsto ax+c \bmod m$ has a fixed point mod m, i.e., an integer v such that $av+c = v \bmod m$.

coef sum	β basis	powers	scratch	x basis
$1 \leftarrow$	$\beta =$	β	$= 2+3x$	$= 2+3x$
$2 \leftarrow$	$3+4\beta =$	β^2	$= 4+12x+9x^2$	$= 1+2x$
$1 \leftarrow$	$2+4\beta =$	β^3	$= 2+7x+6x^2$	$= 2x$
$0 \leftarrow$	$2+3\beta =$	β^4	$= 4x+6x^2$	$= 3+4x$
$3 \leftarrow$	$4+4\beta =$	β^5	$= 6+17x+12x^2$	$= 2+2x$
$2 \leftarrow$	$2 =$	β^6	$= 4+10x+6x^2$	$= 2$
$2 \leftarrow$	$2\beta =$	β^7	$= 4+6x$	$= 4+x$
$4 \leftarrow$	$1+3\beta =$	β^8	$= 8+14x+3x^2$	$= 2+4x$
$2 \leftarrow$	$4+3\beta =$	β^9	$= 4+14x+12x^2$	$= 4x$
$0 \leftarrow$	$4+\beta =$	β^{10}	$= 8x+12x^2$	$= 1+3x$
$1 \leftarrow$	$3+3\beta =$	β^{11}	$= 2+9x+9x^2$	$= 4+4x$
$4 \leftarrow$	$4 =$	β^{12}	$= 8+20x+12^2$	$= 4$
$4 \leftarrow$	$4\beta =$	β^{13}	$= 8+12x$	$= 3+2x$
$3 \leftarrow$	$2+\beta =$	β^{14}	$= 16+48x+36x^2$	$= 4+3x$
$4 \leftarrow$	$3+\beta =$	β^{15}	$= 8+18x+9x^2$	$= 3x$
$0 \leftarrow$	$3+2\beta =$	β^{16}	$= 6x+9x^2$	$= 2+x$
$2 \leftarrow$	$1+\beta =$	β^{17}	$= 4+8x+3x^2$	$= 3+3x$
$3 \leftarrow$	$3 =$	β^{18}	$= 4+15x+9x^2$	$= 3$
$3 \leftarrow$	$3\beta =$	β^{19}	$= 6+9x$	$= 1+4x$
$1 \leftarrow$	$4+2\beta =$	β^{20}	$= 2+11x+12x^2$	$= 3+x$
$3 \leftarrow$	$1+2\beta =$	β^{21}	$= 6+11x+3x^2$	$= x$
$0 \leftarrow$	$1+4\beta =$	β^{22}	$= 2x+3x^2$	$= 4+2x$
$4 \leftarrow$	$2+2\beta =$	β^{23}	$= 8+16x+6x^2$	$= 1+x$
$1 \leftarrow$	$1 =$	β^{24}	$= 2+5x+3x^2$	$= 1$

Thus the powers of β generate all the nonzero elements of G_{25}, so β is primitive. Now we need a method of turning an element of the form $a+b\beta$ into an element of $\mathbb{Z}_5$, since we want our pseudorandom sequence to have values in $\{0, 1, 2, 3, 4\}$. The simplest one is to add coefficients, so that $4+2\beta$ becomes $4+2 = 1 \bmod 5$. This explains the first two columns of the table of powers of β.

This construction is nice, in theory, but rather cumbersome. It is possible to find a linear recurrence relation that will do the same thing. We have already discovered that the polynomial of lowest degree of which β is a root is $z^2 - 4z - 3$, that is $\beta^2 - 4\beta - 3 = 0$ in G_{25}, or equivalently, $\beta^2 = 4\beta + 3$. Thus, for any three consecutive powers of β we have

$$\beta^{k+2} = 4\beta^{k+1} + 3\beta^k.$$

Now we just sum the coefficients (in the β basis) of this relation to get the recurrence relation

$$y_n = (4y_{n-1} + 3y_{n-2}) \bmod 5.$$

Starting with the values $y_0 = 4$ and $y_1 = 1$ gives the data from the table above, or the pseudorandom sequence

$$1\ 2\ 1\ 0\ 3\ 2\ 2\ 4\ 2\ 0\ 1\ 4\ 4\ 3\ 4\ 0\ 2\ 3\ 3\ 1\ 3\ 0\ 4\ 1.$$

Notice that it will not repeat until both seeds y_0 and y_1 simultaneously repeat a previous pair of values. Also notice that any two nonzero y_0 and y_1 will give you the same sequence, just starting at a different point.

A.2 Testing for Nonrandomness

We briefly describe two tests of an RNG sequence to test for nonrandomness (that is, the tests can show only that the sequence is nonrandom, not that the sequence is in fact random). For more tests and discussion, see [Knu81].

A.2.1 Chi-Square Test

The chi-square test is a very general statistical test of the hypothesis that random outcomes are occurring according to some specified distribution. Suppose the possible outcomes of an experiment are partitioned into events E_1, E_2, …, E_k. Further suppose, according to the hypothesis, that the probability of event E_i is p_i, $p_i > 0$, $\sum_1^k p_i = 1$. Now let N trials of the experiment be performed; the expected number of outcomes of E_i is p_iN. Suppose the actual number of outcomes is Y_i. Then

$$V = \sum_{i=1}^{k} \frac{(Y_i - p_iN)^2}{p_iN}$$

is a chi-square random variable with $k-1$ *degrees of freedom* (DOF). The chance that V deviates from 0 is predicted by a chi-square table.

For example, in a sequence of two coin tosses, the probability that HH occurs is 0.25, the same for HT, TH, and TT. Now for 20 trials of the experiment (of tossing a coin twice) suppose we get the following

	$p = 0.01$	$p = 0.05$	$p = 0.25$	$p = 0.50$	$p = 0.75$	$p = 0.95$	$p = 0.99$
$\nu = 1$	0.00016	0.00393	0.1015	0.4549	1.323	3.841	6.635
$\nu = 2$	0.02010	0.1026	0.5753	1.386	2.773	5.991	9.210
$\nu = 3$	0.1148	0.3518	1.213	2.366	4.108	7.815	11.34
$\nu = 4$	0.2971	0.7107	1.923	3.357	5.385	9.488	13.28
$\nu = 5$	0.5543	1.1455	2.675	4.351	6.626	11.07	15.09
$\nu = 6$	0.8720	1.635	3.455	5.348	7.841	12.59	16.81
$\nu = 7$	1.239	2.167	4.255	6.346	9.037	14.07	18.48
$\nu = 8$	1.646	2.733	5.071	7.344	10.22	15.51	20.09
$\nu = 9$	2.088	3.325	5.899	8.343	11.39	16.92	21.67
$\nu = 10$	2.558	3.940	6.737	9.342	12.55	18.31	23.21
$\nu = 11$	3.053	4.575	7.584	10.34	13.70	19.68	24.73
$\nu = 12$	3.571	5.226	8.438	11.34	14.84	21.03	26.22
$\nu = 15$	5.229	7.261	11.04	14.34	18.25	25.00	30.58
$\nu = 20$	8.260	10.85	15.45	19.34	23.83	31.41	37.57
$\nu = 30$	14.95	18.49	24.48	29.34	34.80	43.77	50.89
$\nu = 50$	29.71	34.76	42.94	49.33	56.33	67.50	76.15
$\nu > 30$	$\nu + \sqrt{2\nu}\, x_p + \frac{2}{3}x_p^2 - \frac{2}{3} + O(1/\sqrt{\nu})$						
$x_p =$	-2.33	-1.64	$-.675$	0.00	0.675	1.64	2.33

Table A.1. Chi-square table. Table value x at row ν and column p means that p is the probability a chi-square random variable with ν DOF is $\leq x$.

outcomes: HH 2, HT 5, TH 3, TT 10. How likely is this outcome? The chi-square random variable in this case is

$$V = \frac{(2-0.25\times 20)^2}{0.25\times 20} + \frac{(5-0.25\times 20)^2}{0.25\times 20} + \frac{(4-0.25\times 20)^2}{0.25\times 20} + \frac{(10-0.25\times 20)^2}{0.25\times 20} = 7.$$

According to the chi-square table with three degrees of freedom $\nu = 3$, the probability of this value or less is about 0.906 (interpolating in the table). Put differently, the probability of this value or more is 0.094 or 9.4%. Thus, at the 90% level, this outcome fails the test, but at the 95% level it passes.

A.2.2 Kolmogorov–Smirnov Test

The Kolmogorov–Smirnov test is a general statistical test designed to test the hypothesis that a given set of samples is from a specific distribution. Say that the samples are $x_1, x_2, \ldots, x_N$ and it is desired to test whether they come from a distribution with cdf F. Compute the *empirical cumulative distribution function*

$$\hat{F}_N(t) = \frac{\#\{i : x_i \leq t\}}{N}$$

(the fraction of the number of samples x_i that are less than or equal to t). According to the Glivinko–Cantelli theorem from probability theory (see [Chu74]), if the x_i's are really independent samples from F, then

$$\lim_{N\to\infty} \sup_t |F_N(t) - F(t)| = 0.$$

This says that the biggest difference between F_N and F shrinks to 0 as N, the number of samples, tends to infinity. Let

$$D_N = \sup_t |F_N(t) - F(t)|,$$

as illustrated in Figure A.2, where F is the cdf for the uniform distribution. Under the assumption that the samples X_i are independent and with common distribution F, the distribution of D_N is known and can be tabulated.

In the context of a test for nonuniformity of an RNG, the target distribution is the uniform distribution on $[0,1]$; thus $F(x) = x$. Table A.2 gives the acceptance limits for the Kolmogorov–Smirnov test. To see how to use this, consider the following example.

If F_n is constructed using $n = 12$ samples, then the chance that D_{12} exceeds 0.38 is 5%. In this example we want to test whether the 12 data points,

.45	.22	.06	.93	.91	.61	.33	.80	.95	.74	.88	.21

are from a $U(0,1)$ random number generator at the 95% confidence level.

The first task is to form the cdf of the data, by summing all probabilities less than or equal to x as a function of x. The first row of the following matrix is the sorted data values x; the second is the sum of the number of data values less than x divided by the total number of values, 12.

.06	.21	.22	.33	.45	.61	.74	.80	.88	.91	.93	.95
.083	.166	.25	.333	.416	.50	.583	.666	.75	.833	.916	1

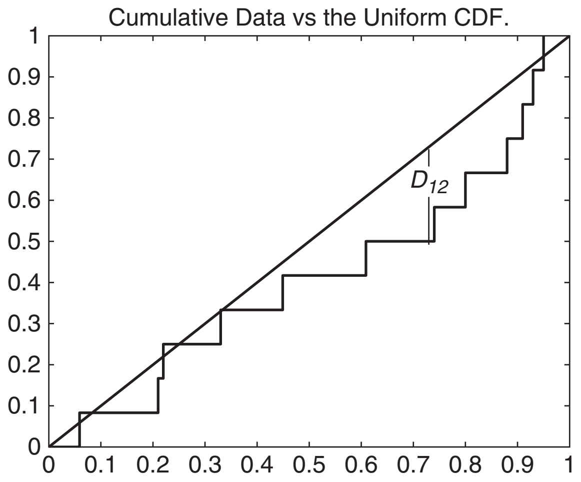


Fig. A.2. Twelve data point empirical CDF vs the uniform cdf.

see Figure A.2. The maximum vertical distance between the two is $D_{12} = 0.239$. Note that this occurs at $x = 0.739$, just before the 0.74 data point. From the line $n = 12$ of Table A.2, with probability 0.2, D_{12} could be as high as 0.30 with the data in fact coming from $U(0, 1)$. Therefore the data passes the test.

As another example, we use the Kolmogorov–Smirnov test at a level of $\alpha = 0.05$ to test the linear congruential RNG of the form

$$x_{n+1} = 1 + 3x_n \bmod p,$$

for $p = 11$ and $p = 229$ and $p = 104729$, each with 1000 samples and all of them with a starting seed of $x_0 = 3$.

For the modulus $p = 11$, clearly the maximum cycle length is 11 (the actual cycle length is 5), and thus in 1000 samples it will run through its cycle many (200) times. We expect poor performance, and we compute that $D_{1000} = 0.3454$, which is much larger than the threshold of $0.043 = 1.36/\sqrt{1000}$, and thus this RNG fails the test (no surprise).

For the modulus $p = 229$, again the generator will run through its cycle several times, but not as many as with $p = 11$. Thus $D_{1000} = 0.1461$ reflects this fact, but still fails the test.

Finally, with the modulus $p = 104729$ we obtain $D_{1000} = 0.0241$, which is significantly below the threshold of 0.043, and thus this RNG passes the Kolmogorov–Smirnov test. However, this doesn't mean that the sequence is actually random, just that we cannot conclude that it isn't random.

	α			
	$p = 0.20$	$p = 0.10$	$p = 0.05$	$p = 0.01$
$n = 1$	0.90	0.95	0.98	0.99
$n = 2$	0.68	0.78	0.84	0.93
$n = 3$	0.56	0.64	0.71	0.83
$n = 4$	0.49	0.56	0.62	0.73
$n = 5$	0.45	0.51	0.56	0.67
$n = 6$	0.41	0.47	0.52	0.62
$n = 7$	0.38	0.44	0.49	0.58
$n = 8$	0.36	0.41	0.46	0.54
$n = 9$	0.34	0.39	0.43	0.51
$n = 10$	0.32	0.37	0.41	0.49
$n = 11$	0.31	0.35	0.39	0.47
$n = 12$	0.30	0.34	0.38	0.45
$n = 15$	0.27	0.30	0.34	0.40
$n = 20$	0.23	0.26	0.29	0.35
$n = 30$	0.19	0.22	0.24	0.29
$n = 35$	0.18	0.21	0.23	0.27
$n = 40$	0.17	0.19	0.21	0.25
$n = 45$	0.16	0.18	0.20	0.24
$n > 45$	$\frac{1.07}{\sqrt{n}}$	$\frac{1.22}{\sqrt{n}}$	$\frac{1.36}{\sqrt{n}}$	$\frac{1.63}{\sqrt{n}}$

Table A.2. Kolmogorov–Smirnov acceptance limits.
$D_n = \|F_n(x) - F(x)\|_\infty$, table value $d_{n,\alpha}$ means $\Pr(D_n \geq d_{n,\alpha}) = \alpha$.

Problems: Appendix A

In this set of problems we want to show that if the random number generator is linear congruential, $x_{n+1} = ax_n + c \mod m$, then it is possible to find the parameters from the sequence of random integers it produces. For this, it is helpful to have a tool from elementary number theory, the *Euclidean algorithm.*

Theorem 2 *Let d be the greatest common divisor of the integers g and h. There exist integers x and y such that $xg + yh = d$.*

Solutions x and y can be found by the division algorithm as follows. Assuming $g > h$, divide g by h to get a quotient and remainder

$$g = q_1 h + r_1.$$

Now do the same with h and r_1:

$$h = q_2 r_1 + r_2.$$

Continue this by shifting the r's to the left, that is,

$$\begin{aligned} r_1 &= q_3 r_2 + r_3, \\ r_2 &= q_4 r_3 + r_4, \\ \vdots\ &\ \ \vdots \\ r_{k-2} &= q_k r_{k-1} + r_k, \\ r_{k-1} &= q_{k+1} r_k + d, \end{aligned}$$

stopping when the remainder becomes d (carrying the division out one more time gives a remainder of 0, hence determining d). Now work back up: In the last equation, substitute for r_k from the previous line

$$d = r_{k-1} - q_{k+1}\left(r_{k-2} - q_k r_{k-1}\right).$$

Then substitute for r_{k-1} from the previous line, and so on.

Note that x and y are not unique.

Note that if the LCRNG is "degenerate," then the parameters are not necessarily unique. For example, the generator $x' = (2x + 1)\%4$ produces $x_0 = 0, 1, 3, 3, \ldots$, at the same time the generator $x' = (4x + 7)\%16$ produces $x_0 = 3, 3, 3, \ldots$. However, if we find either set of parameters or some other one altogether that will give a constant stream of 3's, then at least as far as predicting future numbers is concerned, it does not matter.

1. (2) Assume that m is known and c is 0. Use two successive return values x_{n+1} and x_n to find a.

2. (2) Assume that both m and c are known (but c is not necessarily 0). Use two successive return values x_{n+1} and x_n to find a.

3. (3) Assume that both m and a are known. Use two successive return values x_{n+1} and x_n to find c.

4. (4) Assume that only m is known. Use three successive return values to find a and c.

5. (5) What can be done if none of a, c, and m are known?

B

Perron–Frobenius Theorem

The version of the Perron–Frobenius theorem that we prove here is not the most general version. Our version is specialized to row-stochastic matrices, so it applies directly to the transition matrix for a Markov chain. The more general version requires only that the matrix P be nonnegative, and concludes that there is some eigenvalue $\lambda > 0$ for P such that the corresponding eigenvector is positive and all other eigenvalues have magnitude smaller than λ.

Theorem B.1. *(Perron–Frobenius) Let P be an $n \times n$ row stochastic matrix with $P_{i,j} \geq \delta > 0$ for all i, j. Then P has 1 as an eigenvalue whose corresponding normalized left eigenvector π^T is strictly positive. Furthermore, all the other eigenvalues of P are strictly smaller than one in modulus and*

$$\lim_{n\to\infty} P^n = \mathbb{1}\pi^T,$$

where $\mathbb{1}$ is the (column) vector of all ones. In particular, this means that for all probability vectors α,

$$\lim_{n\to\infty} \alpha P^n = \pi.$$

B.1 Proof of Perron–Frobenius

Let P be a row stochastic matrix and let $\mathbb{1}$ be the (column) vector of all ones. Then $P\mathbb{1} = \mathbb{1}$. Furthermore, if $f^T\mathbb{1} = 0$ then

$$(P^T f)^T \mathbb{1} = f^T(P\mathbb{1}) = f^T\mathbb{1} = 0,$$

so P^T has the space

$$\mathcal{S} = \{v : v^T \mathbb{1} = 0\}$$

as an invariant subspace.

Define $\text{osc}(f) = \max f - \min f$, where $f \in \mathbb{R}^N$.

We restrict our attention to the case that P satisfies the condition

$$P_{i,j} \geq \delta > 0 \quad \forall i, j. \tag{B.1}$$

This condition isn't as restrictive as it might appear. If the associated Markov chain is irreducible (it is possible to go from any state to any other state) and finite, then some power of the transition matrix P will satisfy this condition. Thus, we will assume this condition.

Proposition B.2. *If P satisfies condition (B.1), then for all f,*

$$osc(Pf) \leq (1 - 2\delta) osc(f).$$

Proof. Assume that f_a is the smallest component of f and f_b is the largest, and consider the ith component of Pf,

$$\begin{aligned}(Pf)_i &= p_{i1} f_1 + p_{i2} f_2 + \cdots + p_{iN} f_N \\ &\leq p_{ia} f_a + (p_{i1} + \cdots + p_{i,a-1} + p_{i,a+1} + \cdots + p_{iN}) f_b \\ &\leq \delta f_a + (1 - \delta) f_b.\end{aligned}$$

Similarly,

$$(Pf)_i \geq (1 - \delta) f_a + \delta f_b.$$

Thus for every component of Pf,

$$\begin{aligned}\max(Pf)_i - \min(Pf)_i &\leq [\delta f_a + (1 - \delta) f_b] - [(1 - \delta) f_a + \delta f_b] \\ &= (1 - 2\delta)(f_b - f_a).\end{aligned}$$

Proposition B.3. *Let c_j^n denote the jth column of P^n. If P satisfies condition (B.1), then c_j^n tends to a constant vector as $n \to \infty$, that is, for some constant π_j,*

$$\lim_{n \to \infty} c_j^n = \pi_j \mathbb{1},$$

where $\mathbb{1}$ is the vector of 1's. In particular, letting $\pi = [\pi_1 \ \ \pi_2 \ \ \ldots \ \ \pi_N]^T$, we have

$$\max_i \|P_{i,j}^n - \pi_j\| \leq C(1 - 2\delta)^n \quad \forall j$$

for some constant $C > 0$. Hence $\lim_{n \to \infty} P^n = \mathbb{1}\pi^T$. The vector π is the invariant distribution of P, i.e.,

$$\pi^T P = \pi^T.$$

Proof. With c_j^n as above, we see that

$$\operatorname{osc}(c_j^{n+1}) = \operatorname{osc}(Pc_j^n) \leq (1 - 2\delta)\operatorname{osc}(c_j^n);$$

thus $c_j^n \to \pi_j \mathbb{1}$ as claimed. This implies that π is a probability vector, since P^n is row stochastic for all n. Further, $\pi^T P = \pi^T$, so $\pi^T(P^n - \mathbb{1}\pi^T) = 0$ for all n.

As $n \to \infty$ the convergence above is at least geometric with rate equal to $1 - 2\delta$.

Clearly P has 1 as an eigenvalue, since $P\mathbb{1} = \mathbb{1}$. Thus, what we wish to show is that under the condition (B.1), all the other eigenvalues of P have modulus strictly smaller than one.

Proposition B.4. *For all $f \in \mathcal{S}$, $(P^T)^n f \to 0$ (in fact, geometrically fast).*

Proof. Since $\mathcal{S}$ is an invariant subspace for P^T, if we denote by $\widehat{P^T}$ the restriction of P^T to $\mathcal{S}$, we know that $\widehat{P^T} f = P^T f$ for all $f \in \mathcal{S}$. But then

$$(\widehat{P^T})^n f \to \pi \mathbb{1}^T f = 0,$$

since $f \in \mathcal{S}$.

Thus, all the eigenvalues of $\widehat{P^T}$ are strictly smaller than one in modulus. However, the set of eigenvalues of P^T is equal to $\{1\}$ union the set of eigenvalues of $\widehat{P^T}$ (since $\mathcal{S}$ is invariant for P^T). Thus, P^T has only one eigenvalue of modulus 1; all others have modulus strictly smaller than one. However, the eigenvalues of P are the same as those of P^T. Thus, P has 1 as an eigenvalue with all the rest having modulus strictly smaller than 1.

An Observation

Let 1, μ_2, μ_3, …, μ_n be the eigenvalues of P, in order of decreasing complex modulus; of course $|\mu_2| < 1$. Let π, r_2, r_3, …, r_n be a Jordan basis, so that

$$P = C^{-1} J C,$$

where J is the Jordan form for P and C is the matrix whose rows are the Jordan basis vectors.

Proposition B.5. *The first column of C^{-1} is the vector $\mathbb{1}$.*

Proof. Let $c = [\,\bar{c}_1 \quad \bar{c}_2 \quad \ldots \quad \bar{c}_n\,]^T$ be the first column of C^{-1}. Since $CC^{-1} = I$, the dot products must yield,

$$c \cdot \pi = 1, \qquad c \cdot r_i = 0, \quad i = 2, \ldots, n.$$

Since $r_i \in \mathcal{S}$, $i = 2, \ldots, n$, this system has the unique solution $c = \mathbb{1}$.

Since J is the Jordan form of P, we have $P^k = C^{-1}J^kC$ for all k. Clearly

$$\lim_{k\to\infty} J^k = L = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix}.$$

So from the proposition above,

$$\lim_{k\to\infty} P^k = C^{-1}LC = \mathbb{1}\pi,$$

where the right member is an outer product.

C
Projects

The best way to learn about Monte Carlo methods is to implement one for a real problem. Over the time we taught a course based on this material, we received many outstanding projects from our students. We include below a short list of suggested topics for projects. Of course the best topic is often inspired by a problem that you select yourself and whose solution is important and meaningful to you.

In general, a completed project should include a written report along with commented source code. To be most useful, the report should include any theory or analysis associated with the project, what the software is supposed to do, and how to use it. The report should include a discussion of validation runs as well as production runs. Be sure to include results and conclusions.

This kind of formal write-up is good practice for "real-world" technical reports. In addition, it will provide you with a document that you can refer back to in the future to remind yourself what you did and why you did it.

Some Ideas for Project Topics

1. A genetic algorithm or simulated annealing solution to some combinatorial optimization problem. Examples problems are:
   a) The TSP or TSP with time windows.
   b) The *maximum permanent* problem.
   c) The *maximum clique* problem. Given a graph, a *clique* is a subgraph which is completely connected (every pair of vertices are connected). The problem is, given a graph, find its maximal clique.

   d) The *job shop scheduling* problem. In one form, for a given machine, a number of jobs are to be processed on the machine. Each job has a processing time, a deadline, and a cost if the deadline is not met. How should the jobs be sequenced to minimize the cost?
   e) The *assignment* problem. Given an $N \times N$ non-negative matrix W of weights, find a subset of elements of W, with exactly one element in each row and in each column, having minimal sum.
   f) The *structure of protein.* The determination of the physical structure of a protein is a very difficult optimization problem. This problem starts with a molecule formed as a sequence of (chemical) amino acids, and attempts to find the three-dimensional structure of the molecule which minimizes its internal energy.
   g) The *Steiner network* problem. Given a graph G, find a minimal subset of arcs such that when all arcs are removed but those, a specified subset of vertices remain connected. This problem is related to providing a service, such as electrical power, to a group of cities.
   h) The *shortest path* problem. This problem can take on many different forms; one dichotomy being discrete or continuous. An example of a continuous problem is finding the "optimal path" to fly an aircraft between two points across mountainous and hostile terrain. Most discrete manifestations of the shortest path problem can be framed in terms of graphs. Thus, given two vertices in a weighted graph, find the path which minimizes the sum of weights. A modern example of this problem arises when an online map service attempts to provide a user with a route between two points in a city.

2. Both genetic algorithms and simulated annealing suffer from the problem of niching. One possible solution is to restart the process whenever such niching is detected. Invent several measures of niching and restarting rules and investigate their efficacy, i.e., when is it better to restart rather than to continue the current run.

3. Investigate the parallelization of some MCMC method. This could be practical (choose some problem, implement a parallel version, run simulations, and compare with a serial version) or theoretical (investigate the features of one/several MCMC method(s) which allow parallelism).

4. Program a more realistic version of a game such as Monopoly.

5. Two examples of possible *financial mathematics* topics:

   a) Verify "put-call parity". Let S be the stock price, X be the strike price, T be the time to expiration, and r be the risk-free rate. Further, let P be the cost of a put option and C be the cost of a call option with those given parameters (S, X, T, and r) and any volatility. *Put-call parity* asserts that

   $$S + P - C = Xe^{-rT}.$$

   Note that since this is assumed to hold for any volatility, all risk has been hedged away.
   Test if put-call parity holds by making a choice of the parameters and estimating P and C by Monte Carlo. Do this for several choices of parameters. Of course, the Monte Carlo results will only be approximate by the nature of the method. You can improve the results by arranging your program to compute both P and C in the same Monte Carlo run, that is, using the same sequence of random numbers. This technique is called *variance reduction.*

   b) How much should the following option cost? The current stock price of ABC is $S_1 = \$30$ and that of XYZ is $S_2 = \$32$. The option states that if, 30 days from now, the price of ABC is below \$29, then the payoff is $\max(S_1 - S_2, 0)$ (those being the respective stock prices at the expiration on day 30). In other words, S_2 must be below S_1 and both must be below \$29 for the option to payout. The respective volatilities are $v_1 = 0.3$ and $v_2 = 0.4$, and the risk-free rate is $r = 0.3\%$.

6. Simulate some version of a simple artificial ecosystem. As an example:

   Artificial life simulation, everything takes place in a 100 meter by 100 meter square.
   Laws of physics: each season enough sunlight falls for exactly 100 new plants to grow, randomly distributed. There is space for 1000 eaters. An eater coming within 2 meters of a plant eats it all and gets 100 calories of energy. The most an eater can store is 200 calories. It takes 1 calorie to move 2 meters and 20 to have an offspring. Eaters live at most 20 seasons.

Invent a 4 gene eater in which the genes code for movement and reproduction control. Simulate 100 time periods per season. In one time period an eater can move and eat or reproduce.

7. Use MCMC to convert a game of partial information to one of stochastic, but full information (as in Section 3.7.1). As an example:

   (Mini-hearts, using Monte Carlo to convert to complete information.) Optimal "Hearts" play, assume 3 cards per suit (K, Q, J), 4 suits as usual. Assume 4 players.
   Rules: cards are shuffled and completely dealt out, the hand with the club-J leads (begins a trick) by playing it; play means to place a card face up in the center of the table. Play proceeds clockwise from the hand that leads. A play must follow suit if possible, otherwise any card may be played. After all have played to a trick, the one who placed the highest card of the suit lead wins the trick, and then leads to the next trick.
   Winning: taking a Heart on a trick counts 1 point, taking the spade-Q counts 3 points. Player with the lowest count wins.
   Strategy: To make a play that is the best possible, use Monte Carlo simulation to distribute the, as yet, unseen cards into the opponents hands. Given such a sample, use a complete information analysis to decide the best play for that sample. Histogram, weight, and select the overall (stochastically) best play.

8. *Fractal inverse problem*

   Let $\mathcal{W} = \{w_1, w_2, \dots, w_n\}$ be a finite set of affine maps of the unit square $I = [0,1] \times [0,1]$ into itself, that is, maps of the form

$$w_k(\mathbf{x}) = \begin{bmatrix} a_k & b_k \\ c_k & d_k \end{bmatrix} \mathbf{x} + \begin{bmatrix} e_k \\ f_k \end{bmatrix}.$$

   Here $\mathbf{x}$ denotes the 2D vector $\begin{bmatrix} x \\ y \end{bmatrix}$.
   Associated with every such collection $\mathcal{W}$ is the *attractor* A, that is the unique compact subset $A \subset I$ characterized by the self-covering property

$$A = \bigcup_{i=1}^{n} w_i(A).$$

   For the proof of the existence and uniqueness of an attractor see [Bar88]. It is, however, easy to obtain points in A. Obviously the fixed point $\mathbf{x}_i^*$ of each map w_i in $\mathcal{W}$ belongs to A. Further, this

equation provides that if $\mathbf{x} \in A$ then also $w_i(\mathbf{x}) \in A$ for each $i = 1, 2, \ldots, n$. It follows that for every composition $f = w_{i_1} \circ w_{i_2} \circ \cdots \circ w_{i_k}$, where $i_j \in \{1, \ldots, n\}$ for all $j = 1, \ldots, k$, if $\mathbf{x} \in A$ then also $f(\mathbf{x}) \in A$. It is in this way that $\mathcal{W}$ is an iterated function system (IFS).
Moreover the observation above provides a method to visualize an attractor on a computer screen. Starting from the fixed point $\mathbf{x}^*$ of map w_1, choose $i_1 \in \{1, 2, \ldots, n\}$ at random and plot $\mathbf{x}_1 = w_{i_1}(\mathbf{x}^*)$. Now repeat this step with $\mathbf{x}_1$ in place of $\mathbf{x}^*$, set $x_2 = w_{i_2}(\mathbf{x}_1)$, and plot $\mathbf{x}_2$. Continuing this way, we set $\mathbf{x}_{n+1} = w_{i_n}(\mathbf{x}_n)$ and repeat until some large number of points, say something like 10,000, have been plotted. This construction is known as the *Random Iteration algorithm* for constructing the attractor. Typically the attractor of an IFS is a fractal set.
As an example, the Sierpinski gasket is the attractor of the IFS

$$\begin{aligned}
w_1(\mathbf{x}) &= \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \mathbf{x}, \\
w_2(\mathbf{x}) &= \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1/2 \\ 0 \end{bmatrix}, \\
w_3(\mathbf{x}) &= \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1/4 \\ 1/2 \end{bmatrix}.
\end{aligned}$$

The *Inverse Fractal problem* is that of finding an IFS whose attractor is the given set, or as close to the given set as possible.

References

[Ash06] Ashlock, D., *Evolutionary Computation for Modeling and Optimization*, Springer-Verlag, New York, 2006.

[BFM97] Bäck, T., Fogel, D., Michalewicz, Z., *Handbook of Evolutionary Computation*, Oxford, 1997.

[Bar88] Barnsley, M.F., *Fractals Everywhere*, Academic Press, 1988.

[BH62] Hull, T.E.,Dobell, A.R., "Random Number Generators," *SIAM Review*, **4**, 230–254, 1962.

[BFS83] Bratley, P., Fox, B., and Scrage, L., *A Guide to Simulation*, Springer-Verlag, New York, N.Y., 1983.

[Bre99] Brémaud, P., *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*, Springer-Verlag, New York, 1999.

[Chu74] Chung, K.L., *A course in probability theory*, Academic Press, New York, 1974.

[BHR93] de Boor, C., Höllig, K., Riemenschneider, S., *Box Splines*, Springer-Verlag, New York, 1993.

[DS84] Doyle, P., Snell, J.L., *Random walks and electric networks*, Carus Mathematical Monographs, MAA, 1984.

[Gol89] Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusets, 1989.

[G92] Gillman, Leonard, "The Car and the Goats," *American Mathematical Monthly* Vol 99, 3–7, 1992.

[GH05] Givens, G and Hoeting, J., *Computational Statistics*, Wiley-Interscience, Hoboken, New Jersey, 2005.

[Gri03] Grimaldi, R., *Discrete and Combinatorial Mathematics*, Addison Wesley, 2003.

[Haj88] Hajek, B., "Cooling schedules for optimal annealing," *Math. Operat. Res.*, **13**, No. 2, 311-329.

[HH64] Hammersley, J.M., Handscomb, D.C., *Monte Carlo Methods*, Wiley, New York, 1964.

[HL96] Hanselman, D., Littlefield, B., *Mastering Matlab*, Prentice-Hall, Upper Saddle River, N.J., 1996.

[Has70] Hastings, W.K., "Monte Carlo Sampling Methods Using Markov Chains and their Applications," *Biometrica* **57** (1970), 97-109.

[HT01] Hogg, R.V., Tanis, E.A., *Probability and Statistical Inference*, Prentice Hall, Upper Saddle River, N.J., 2001.

[Hul03] Hull, John, C., *Options, Futures, and Other Derivatives, Fifth Edition*, Prentice Hall, Upper Saddle River, N.J., 2003.

[ISAC95] Isaac, R., *The Pleasures of Probability*, Springer-Verlag, New York, 1995.

[KT73] Kahneman, D., Tversky, A., "On the psychology of prediction," *Psychological Review* **80** (1973), 237-251.

[KW86] Kalos, M.H., Whitlock, P.A., *Monte Carlo Methods*, Wiley-Interscience, 1986.

[Kel56] Kelly, J.L., Jr., "A New Interpretation of Information Rate," Bell System Technical Journal, 917–926, July 1956.

[KGV83] Kirkpatrick, S., Gelatt, C., Vecchi, M., "Optimization by simulated annealing," *Science* **220**, 671–680, 1983.

[Kit69] Kittel, C., *Thermal Physics*, John Wiley and Sons, New York, 1969.

[Knu81] Knuth, D., *Seminumerical Algorithms*, The Art of Computer Programming, Vol. 2, Addison Wesley, 1981.

[Liu01] Liu, J., *Monte Carlo Strategies in Scientific Computing*, Springer-Verlag, New York, 2001.

[Mad02] Madras, N., *Lectures on Monte Carlo Methods*, American Mathematical Society, Providence, R.I., 2002.

[Mar72] Marsaglia, G., "The Structure of Linear Congruential Sequences," *App. of Num. Theory to N.A.*, ed. S. Zaremba, Academic Press, 249-286.

[MRRTT] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., "Equations of state calculations by fast computing machines," *J. of Chem. Phys.*, **21**, 1087–1091, 1953.

[Nied] Niederetter, H., "Random Number Generation and Quasi-Monte Carlo Methods," CBMS-NSF Regional Conference Series in Applied Mathematics **63**, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[OBH04] Ohlmann, J., Bean, J., Henderson, S., "Convergence in Probability of Compressed Annealing", *Math of OR* **29**, Issue 4, 837–860, 2004.

[Pou05] Poundstone, W., *Fortunes Formula*, Hill and Wang, New York, N.Y., 2005.

[PW96] Propp, J. and Wilson, D., "Exact sampling with coupled Markov chains and applications to statistical mechanics", *Random Structures and Algorithms*, **9** 1&2 (1996), 223-252.

[Rip87] Ripley, B.D., *Stochastic Simulation*, John Wiley and Sons, 1987.

[Ross96] Ross, S., *Stochastic Processes, Second Edition*, John Wiley & Sons, New York, N.Y., 1996.

[Ross05] Ross, S., *A first course in probability (7th ed)*, Prentice Hall, Upper Saddle River, N.J., 2005.

[Rub81] Rubenstein, R. Y., *Simulation and the Monte Carlo Method*, John Wiley & Sons, New York, N.Y., 1981.

[Tho97] Thorpe, E. O., "The Kelly Criterion in Blackjack, Sports Betting, and the Stock Market,", paper presented at the 10th International Conference on Gambling and Risk Taking, 1997.

[Yak77] Yakowitz, S. J., *Computational Probability and Simulation*, Addison-Wesley, 1977.

[Asa05] Asai, S. "Sons or daughters?", in *Social Organization of the Rufous Vanga: The Ecology of Vangas–Birds Endemic to Madagascar* edited by S. Yamagishi, Trans Pacific Press, 2005.

List of notation

Symbol	*Page*	*Meaning*
$U(a, b)$	15	uniform distribution on $[a, b]$
$\sim$	16	is a sample from
$\mathbb{E}(X)$	17	expectation of X
μ	17	expected value
$\mathrm{var}(X)$	17	variance of X
σ	19	standard deviation
$\Pr(A\|B)$	21	conditional probability of A given B
$\mathrm{cov}(X, Y)$	25	covariance of the rvs X and Y
$a \bmod m$	33	equivalence class modulo m, in range 0 to $(m-1)$
$\mathrm{int}[x]$	56	integer part of x
$\mathrm{frac}(x)$	56	fractional part of x
$b(n, p)$	59	binomial distribution
$E(\lambda)$	64	exponential distribution
$\mathrm{Be}(\alpha, \beta)$	85	beta distribution
$G(\alpha, \lambda)$	89	gamma distribution
$\mathit{\Gamma}(x)$	90	gamma function
N_x	102	Neighborhood of a state
p_{xy}	102	transition probability from x to y
p_t	105	probability vector of a Markov chain
π	106	invariant distribution of a Markov chain
k_B	121	Boltzmann's constant
r_x	182	probability of gamber's ruin
h_x	182	probability of breaking the house

Index