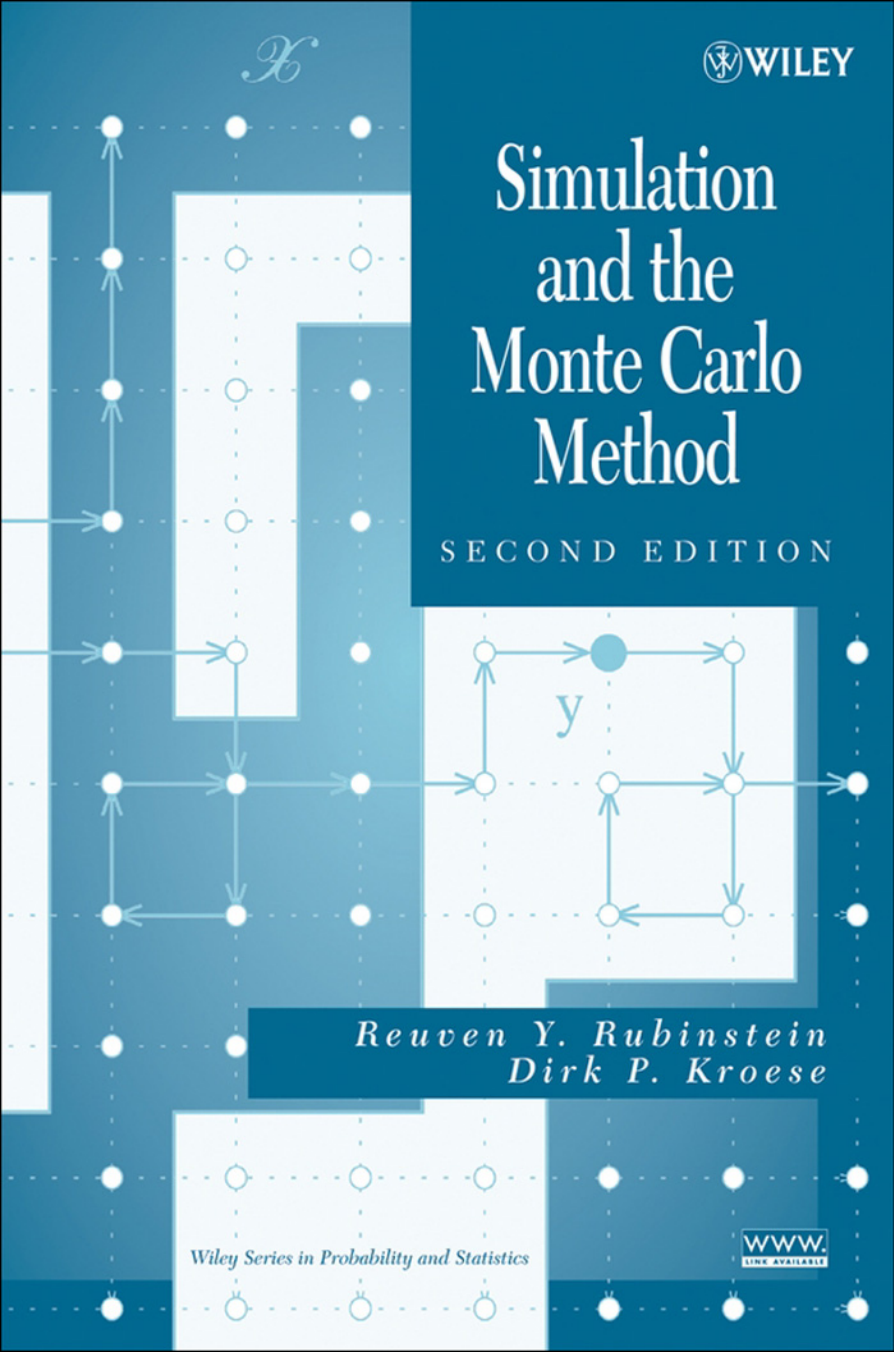

WILEY
Simulation and the Monte Carlo Method
SECOND EDITION
Reuven Y. Rubinstein
Dirk P. Kroese
Wiley Series in Probability and Statistics
WWW.
LINK AVAILABLE

SIMULATION AND THE MONTE CARLO METHOD

Second Edition

Reuven Y. Rubinstein
Technion

Dirk P. Kroese
University of Queensland

WILEY-INTERSCIENCE
A John Wiley & Sons, Inc., Publication

This Page Intentionally Left Blank

SIMULATION AND THE MONTE CARLO METHOD

THE WILEY BICENTENNIAL–KNOWLEDGE FOR GENERATIONS

Each generation has its unique needs and aspirations. When Charles Wiley first opened his small printing shop in lower Manhattan in 1807, it was a generation of boundless potential searching for an identity. And we were there, helping to define a new American literary tradition. Over half a century later, in the midst of the Second Industrial Revolution, it was a generation focused on building the future. Once again, we were there, supplying the critical scientific, technical, and engineering knowledge that helped frame the world. Throughout the 20th Century, and into the new millennium, nations began to reach out beyond their own borders and a new international community was born. Wiley was there, expanding its operations around the world to enable a global exchange of ideas, opinions, and know-how.

For 200 years, Wiley has been an integral part of each generation's journey, enabling the flow of information and understanding necessary to meet their needs and fulfill their aspirations. Today, bold new technologies are changing the way we live and learn. Wiley will be there, providing you the must-have knowledge you need to imagine new worlds, new possibilities, and new opportunities.

Generations come and go, but you can always count on Wiley to provide you the knowledge you need, when and where you need it!

William J. Pesce

WILLIAM J. PESCE
PRESIDENT AND CHIEF EXECUTIVE OFFICER

Peter Booth Wiley

PETER BOOTH WILEY
CHAIRMAN OF THE BOARD

SIMULATION AND THE MONTE CARLO METHOD

Second Edition

Reuven Y. Rubinstein
Technion

Dirk P. Kroese
University of Queensland

WILEY-INTERSCIENCE
A John Wiley & Sons, Inc., Publication

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Rubinstein, Reuven Y.
Simulation and the monte carlo method. — 2nd ed. / Reuven Y. Rubinstein.
Dirk P. Kroese.
p. cm. — (Wiley series in probability and statistics)
Includes index.
ISBN 978-0-470-17794-5 (cloth : acid-free paper)
1. Monte Carlo method. 2. Digital computer simulation. I. Kroese, Dirk P.
II. Title.
QA298.R8 2008
518'.282—dc22 2007029068

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To my friends and colleagues Søren Asmussen and Peter Glynn

— *RYR*

In memory of my parents Albert and Anna Kroese

— *DPK*

This Page Intentionally Left Blank

CONTENTS

This Page Intentionally Left Blank

PREFACE

Since the publication in 1981 of *Simulation and the Monte Carlo Method*, dramatic changes have taken place in the entire field of Monte Carlo simulation. This long-awaited second edition gives a fully updated and comprehensive account of the major topics in Monte Carlo simulation.

The book is based on an undergraduate course on Monte Carlo methods given at the Israel Institute of Technology (Technion) and the University of Queensland for the past five years. It is aimed at a broad audience of students in engineering, physical and life sciences, statistics, computer science and mathematics, as well as anyone interested in using Monte Carlo simulation in his or her study or work. Our aim is to provide an accessible introduction to modern Monte Carlo methods, focusing on the main concepts while providing a sound foundation for problem solving. For this reason, most ideas are introduced and explained via concrete examples, algorithms, and experiments.

Although we assume that the reader has some basic mathematical knowledge, such as gained from an elementary course in probability and statistics, we nevertheless review the basic concepts of probability, Markov processes, and convex optimization in Chapter 1.

In a typical stochastic simulation, randomness is introduced into simulation models via independent uniformly distributed random variables. These random variables are then used as building blocks to simulate more general stochastic systems. Chapter 2 deals with the generation of such random numbers, random variables, and stochastic processes.

Many real-world complex systems can be modeled as discrete-event systems. Examples of discrete-event systems include traffic systems, flexible manufacturing systems, computer-communications systems, inventory systems, production lines, coherent lifetime systems, PERT networks, and flow networks. The behavior of such systems is identified via a

sequence of discrete *events*, which causes the system to change from one *state* to another. We discuss how to model such systems on a computer in Chapter 3.

Chapter 4 treats the statistical analysis of the output data from static and dynamic models. The main difference is that the former do not evolve in time, while the latter do. For the latter, we distinguish between finite-horizon and steady-state simulation. Two popular methods for estimating steady-state performance measures — the batch means and regenerative methods — are discussed as well.

Chapter 5 deals with variance reduction techniques in Monte Carlo simulation, such as antithetic and common random numbers, control random variables, conditional Monte Carlo, stratified sampling, and importance sampling. The last is the most widely used variance reduction technique. Using importance sampling, one can often achieve substantial (sometimes dramatic) variance reduction, in particular when estimating rare-event probabilities. While dealing with importance sampling we present two alternative approaches, called the *variance minimization* and *cross-entropy* methods. In addition, this chapter contains two new importance sampling–based methods, called *the transform likelihood ratio method* and *the screening method* for variance reduction. The former presents a simple, convenient, and unifying way of constructing efficient IS estimators, while the latter ensures lowering of the dimensionality of the importance sampling density. This is accomplished by identifying (screening out) the most important (bottleneck) parameters to be used in the importance sampling distribution. As a result, the accuracy of the importance sampling estimator increases substantially.

We present a case study for a high-dimensional complex electric power system and show that without screening the importance sampling estimator, containing hundreds of likelihood ratio terms, would be quite unstable and thus would fail to work. In contrast, when using screening, one obtains an accurate low-dimensional importance sampling estimator.

Chapter 6 gives a concise treatment of the generic *Markov chain Monte Carlo* (MCMC) method for *approximately* generating samples from an arbitrary distribution. We discuss the classic Metropolis–Hastings algorithm and the Gibbs sampler. In the former, one simulates a Markov chain such that its stationary distribution coincides with the target distribution, while in the latter, the underlying Markov chain is constructed on the basis of a sequence of conditional distributions. We also deal with applications of MCMC in Bayesian statistics and explain how MCMC is used to sample from the Boltzmann distribution for the Ising and Potts models, which are extensively used in statistical mechanics. Moreover, we show how MCMC is used in the simulated annealing method to find the global minimum of a multiextremal function. Finally, we show that both the Metropolis–Hastings and Gibbs samplers can be viewed as special cases of a general MCMC algorithm and then present two more modifications, namely, the *slice* and *reversible jump* samplers.

Chapter 7 focuses on sensitivity analysis and Monte Carlo optimization of simulated systems. Because of their complexity, the performance evaluation of discrete-event systems is usually studied by simulation, and it is often associated with the estimation of the performance function with respect to some controllable parameters. Sensitivity analysis is concerned with evaluating sensitivities (gradients, Hessians, etc.) of the performance function with respect to system parameters. It provides guidance to operational decisions and plays an important role in selecting system parameters that optimize the performance measures. Monte Carlo optimization deals with solving stochastic programs, that is, optimization problems where the objective function and some of the constraints are unknown and need to be obtained via simulation. We deal with sensitivity analysis and optimization of both static and dynamic models. We introduce the celebrated *score function* method for sensitivity analysis, and two alternative methods for Monte Carlo optimization, the so-

called *stochastic approximation* and *stochastic counterpart* methods. In particular, in the latter method, we show how, using a single simulation experiment, one can approximate quite accurately the true unknown optimal solution of the original deterministic program.

Chapter 8 deals with the *cross-entropy* (CE) method, which was introduced by the first author in 1997 as an adaptive algorithm for rare-event estimation using a CE minimization technique. It was soon realized that the underlying ideas had a much wider range of application than just in rare event simulation; they could be readily adapted to tackle quite general combinatorial and multiextremal optimization problems, including many problems associated with learning algorithms and neural computation. We provide a gradual introduction to the CE method and show its elegance and versatility. In particular, we present a general CE algorithm for the estimation of rare-event probabilities and then slightly modify it for solving combinatorial optimization problems. We discuss applications of the CE method to several combinatorial optimization problems, such as the max-cut problem and the traveling salesman problem, and provide supportive numerical results on its effectiveness. Due to its versatility, tractability, and simplicity, the CE method has great potential for a diverse range of new applications, for example in the fields of computational biology, DNA sequence alignment, graph theory, and scheduling. During the past five to six years at least 100 papers have been written on the theory and applications of CE. For more details, see the Web site `www.cemethod.org`; the book by R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning* (Springer, 2004); or Wikipedia under the name *cross-entropy method*.

Finally, Chapter 9 deals with difficult *counting problems*, which occur frequently in many important problems in science, engineering, and mathematics. We show how these problems can be viewed as particular instances of *estimation* problems and thus can be solved efficiently via Monte Carlo techniques, such as importance sampling and MCMC. We also show how to resolve the "degeneracy" in the likelihood ratio, which typically occurs in high-dimensional counting problems, by introducing a particular modification of the classic *MinxEnt* method called *parametric MinxEnt*.

A wide range of problems is provided at the end of each chapter. More difficult sections and problems are marked with an asterisk (*). Additional material, including a brief introduction to exponential families, a discussion on the computational complexity of stochastic programming problems, and sample Matlab programs, is given in the Appendix. This book is accompanied by a detailed *solutions manual*.

REUVEN RUBINSTEIN AND DIRK KROESE

Haifa and Brisbane
July, 2007

This Page Intentionally Left Blank

ACKNOWLEDGMENTS

We thank all who contributed to this book. Robert Smith and Zelda Zabinski read and provided useful suggestions on Chapter 6. Alex Shapiro kindly provided a detailed account of the complexity of stochastic programming problems (Section A.8). We are grateful to the many undergraduate and graduate students at the Technion and the University of Queensland who helped make this book possible and whose valuable ideas and experiments were extremely encouraging and motivating: Yohai Gat, Uri Dubin, Rostislav Man, Leonid Margolin, Levon Kikinian, Ido Leichter, Andrey Dolgin, Dmitry Lifshitz, Sho Nariai, Ben Roberts, Asrul Sani, Gareth Evans, Grethe Casson, Leesa Wockner, Nick Miller, and Chung Chan. We are especially indebted to Thomas Taimre and Zdravko Botev, who conscientiously worked through the whole manuscript, tried and solved all the exercises and provided exceptional feedback. This book was supported by the Australian Research Council under Grants DP056631 and DP055895.

RYR, DPK

This Page Intentionally Left Blank

CHAPTER 1

PRELIMINARIES

1.1 RANDOM EXPERIMENTS

The basic notion in probability theory is that of a *random experiment*: an experiment whose outcome cannot be determined in advance. The most fundamental example is the experiment where a fair coin is tossed a number of times. For simplicity suppose that the coin is tossed three times. The *sample space*, denoted Ω, is the set of all possible outcomes of the experiment. In this case Ω has eight possible outcomes:

$$\Omega = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\},$$

where, for example, HTH means that the first toss is heads, the second tails, and the third heads.

Subsets of the sample space are called *events*. For example, the event A that the third toss is heads is

$$A = \{HHH, HTH, THH, TTH\}.$$

We say that event A *occurs* if the outcome of the experiment is one of the elements in A. Since events are sets, we can apply the usual set operations to them. For example, the event $A \cup B$, called the *union* of A and B, is the event that A or B or both occur, and the event $A \cap B$, called the *intersection* of A and B, is the event that A and B both occur. Similar notation holds for unions and intersections of more than two events. The event A^c, called the *complement* of A, is the event that A does not occur. Two events A and B that have no outcomes in common, that is, their intersection is empty, are called *disjoint* events. The main step is to specify the probability of each event.

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

Definition 1.1.1 (Probability) A *probability* $\mathbb{P}$ is a rule that assigns a number $0 \leqslant \mathbb{P}(A) \leqslant 1$ to each event A, such that $\mathbb{P}(\Omega) = 1$, and such that for any sequence $A_1, A_2, \ldots$ of disjoint events

$$\mathbb{P}(\bigcup_i A_i) = \sum_i \mathbb{P}(A_i) \,. \tag{1.1}$$

Equation (1.1) is referred to as the *sum rule* of probability. It states that if an event can happen in a number of different ways, but not simultaneously, the probability of that event is simply the sum of the probabilities of the comprising events.

For the fair coin toss experiment the probability of any event is easily given. Namely, because the coin is fair, each of the eight possible outcomes is equally likely, so that $\mathbb{P}(\{HHH\}) = \cdots = \mathbb{P}(\{TTT\}) = 1/8$. Since any event A is the union of the "elementary" events $\{HHH\}, \ldots, \{TTT\}$, the sum rule implies that

$$\mathbb{P}(A) = \frac{|A|}{|\Omega|} \,, \tag{1.2}$$

where $|A|$ denotes the number of outcomes in A and $|\Omega| = 8$. More generally, if a random experiment has finitely many and equally likely outcomes, the probability is always of the form (1.2). In that case the calculation of probabilities reduces to counting.

1.2 CONDITIONAL PROBABILITY AND INDEPENDENCE

How do probabilities change when we know that some event $B \subset \Omega$ has occurred? Given that the outcome lies in B, the event A will occur if and only if $A \cap B$ occurs, and the relative chance of A occurring is therefore $\mathbb{P}(A \cap B)/\mathbb{P}(B)$. This leads to the definition of the *conditional probability* of A given B:

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} \,. \tag{1.3}$$

For example, suppose we toss a fair coin three times. Let B be the event that the total number of heads is two. The conditional probability of the event A that the first toss is heads, given that B occurs, is $(2/8)/(3/8) = 2/3$.

Rewriting (1.3) and interchanging the role of A and B gives the relation $\mathbb{P}(A \cap B) = \mathbb{P}(A)\,\mathbb{P}(B \mid A)$. This can be generalized easily to the *product rule* of probability, which states that for any sequence of events $A_1, A_2, \ldots, A_n$,

$$\mathbb{P}(A_1 \cdots A_n) = \mathbb{P}(A_1)\,\mathbb{P}(A_2 \mid A_1)\,\mathbb{P}(A_3 \mid A_1 A_2) \cdots \mathbb{P}(A_n \mid A_1 \cdots A_{n-1}) \,, \tag{1.4}$$

using the abbreviation $A_1 A_2 \cdots A_k \equiv A_1 \cap A_2 \cap \cdots \cap A_k$.

Suppose $B_1, B_2, \ldots, B_n$ is a *partition* of Ω. That is, $B_1, B_2, \ldots, B_n$ are disjoint and their union is Ω. Then, by the sum rule, $\mathbb{P}(A) = \sum_{i=1}^n \mathbb{P}(A \cap B_i)$ and hence, by the definition of conditional probability, we have the *law of total probability*:

$$\mathbb{P}(A) = \sum_{i=1}^n \mathbb{P}(A \mid B_i)\,\mathbb{P}(B_i) \,. \tag{1.5}$$

Combining this with the definition of conditional probability gives *Bayes' rule*:

$$\mathbb{P}(B_j \mid A) = \frac{\mathbb{P}(A \mid B_j)\,\mathbb{P}(B_j)}{\sum_{i=1}^n \mathbb{P}(A \mid B_i)\,\mathbb{P}(B_i)} \,. \tag{1.6}$$

Independence is of crucial importance in probability and statistics. Loosely speaking, it models the lack of information between events. Two events A and B are said to be *independent* if the knowledge that B has occurred does not change the probability that A occurs. That is, A, B independent $\Leftrightarrow \mathbb{P}(A \mid B) = \mathbb{P}(A)$. Since $\mathbb{P}(A \mid B) = \mathbb{P}(A \cap B)/\mathbb{P}(B)$, an alternative definition of independence is

$$A, B \text{ independent} \Leftrightarrow \mathbb{P}(A \cap B) = \mathbb{P}(A)\,\mathbb{P}(B) .$$

This definition covers the case where $B = \emptyset$ (empty set). We can extend this definition to arbitrarily many events.

Definition 1.2.1 (Independence) The events $A_1, A_2, \ldots,$ are said to be *independent* if for any k and any choice of distinct indices $i_1, \ldots, i_k$,

$$\mathbb{P}(A_{i_1} \cap A_{i_2} \cap \cdots \cap A_{i_k}) = \mathbb{P}(A_{i_1})\,\mathbb{P}(A_{i_2}) \cdots \mathbb{P}(A_{i_k}) .$$

Remark 1.2.1 In most cases, independence of events is a model assumption. That is, we assume that there exists a $\mathbb{P}$ such that certain events are independent.

■ **EXAMPLE 1.1**

We toss a biased coin n times. Let p be the probability of heads (for a fair coin $p = 1/2$). Let A_i denote the event that the i-th toss yields heads, $i = 1, \ldots, n$. Then $\mathbb{P}$ should be such that the events $A_1, \ldots, A_n$ are independent, and $\mathbb{P}(A_i) = p$ for all i. These two rules completely specify $\mathbb{P}$. For example, the probability that the first k throws are heads and the last $n - k$ are tails is

$$\begin{aligned} \mathbb{P}(A_1 \cdots A_k A_{k+1}^c \cdots A_n^c) &= \mathbb{P}(A_1) \cdots \mathbb{P}(A_k)\,\mathbb{P}(A_{k+1}^c) \cdots \mathbb{P}(A_n^c) \\ &= p^k (1-p)^{n-k} . \end{aligned}$$

1.3 RANDOM VARIABLES AND PROBABILITY DISTRIBUTIONS

Specifying a model for a random experiment via a complete description of Ω and $\mathbb{P}$ may not always be convenient or necessary. In practice we are only interested in various observations (that is, numerical measurements) in the experiment. We incorporate these into our modeling process via the introduction of *random variables*, usually denoted by capital letters from the last part of the alphabet, e.g., X, $X_1, X_2, \ldots, Y, Z$.

■ **EXAMPLE 1.2**

We toss a biased coin n times, with p the probability of heads. Suppose we are interested only in the number of heads, say X. Note that X can take any of the values in $\{0, 1, \ldots, n\}$. The *probability distribution* of X is given by the *binomial formula*

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \quad k = 0, 1, \ldots, n . \tag{1.7}$$

Namely, by Example 1.1, each elementary event $\{HTH \cdots T\}$ with exactly k heads and $n - k$ tails has probability $p^k(1-p)^{n-k}$, and there are $\binom{n}{k}$ such events.

The probability distribution of a general random variable X — identifying such probabilities as $\mathbb{P}(X = x)$, $\mathbb{P}(a \leqslant X \leqslant b)$, and so on — is completely specified by the *cumulative distribution function* (cdf), defined by

$$F(x) = \mathbb{P}(X \leqslant x), \quad x \in \mathbb{R} .$$

A random variable X is said to have a *discrete* distribution if, for some finite or countable set of values $x_1, x_2, \ldots$, $\mathbb{P}(X = x_i) > 0$, $i = 1, 2, \ldots$ and $\sum_i \mathbb{P}(X = x_i) = 1$. The function $f(x) = \mathbb{P}(X = x)$ is called the *probability mass function* (pmf) of X — but see Remark 1.3.1.

■ **EXAMPLE 1.3**

Toss two fair dice and let M be the largest face value showing. The pmf of M is given by

m	1	2	3	4	5	6	$\sum$
$f(m)$	$\frac{1}{36}$	$\frac{3}{36}$	$\frac{5}{36}$	$\frac{7}{36}$	$\frac{9}{36}$	$\frac{11}{36}$	1

For example, to get $M = 3$, either $(1,3), (2,3), (3,3), (3,2)$, or $(3,1)$ has to be thrown, each of which happens with probability $1/36$.

A random variable X is said to have a *continuous* distribution if there exists a positive function f with total integral 1, such that for all a, b

$$\mathbb{P}(a \leqslant X \leqslant b) = \int_a^b f(u)\, du . \tag{1.8}$$

The function f is called the *probability density function* (pdf) of X. Note that in the continuous case the cdf is given by

$$F(x) = \mathbb{P}(X \leqslant x) = \int_{-\infty}^{x} f(u)\, du ,$$

and f is the derivative of F. We can interpret $f(x)$ as the probability "density" at $X = x$ in the sense that

$$\mathbb{P}(x \leqslant X \leqslant x + h) = \int_x^{x+h} f(u)\, du \approx h\, f(x) .$$

Remark 1.3.1 (Probability Density) Note that we have deliberately used the *same* symbol, f, for both pmf and pdf. This is because the pmf and pdf play very similar roles and can, in more advanced probability theory, both be viewed as particular instances of the general notion of *probability density*. To stress this viewpoint, we will call f in *both* the discrete and continuous case the pdf or (probability) density (function).

1.4 SOME IMPORTANT DISTRIBUTIONS

Tables 1.1 and 1.2 list a number of important continuous and discrete distributions. We will use the notation $X \sim f$, $X \sim F$, or $X \sim \mathsf{Dist}$ to signify that X has a pdf f, a cdf F or a distribution Dist. We sometimes write f_X instead of f to stress that the pdf refers to the random variable X. Note that in Table 1.1, Γ is the gamma function:

$$\Gamma(\alpha) = \int_0^\infty \mathrm{e}^{-x} x^{\alpha-1}\, dx\,, \quad \alpha > 0\,.$$

Table 1.1 Commonly used continuous distributions.

Name	Notation	$f(x)$	$x \in$	Params.
Uniform	$\mathsf{U}[\alpha,\beta]$	$\dfrac{1}{\beta-\alpha}$	$[\alpha,\beta]$	$\alpha<\beta$
Normal	$\mathsf{N}(\mu,\sigma^2)$	$\dfrac{1}{\sigma\sqrt{2\pi}}\,\mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$	$\mathbb{R}$	$\sigma>0,\ \mu\in\mathbb{R}$
Gamma	$\mathsf{Gamma}(\alpha,\lambda)$	$\dfrac{\lambda^\alpha x^{\alpha-1}\mathrm{e}^{-\lambda x}}{\Gamma(\alpha)}$	$\mathbb{R}_+$	$\alpha,\lambda>0$
Exponential	$\mathsf{Exp}(\lambda)$	$\lambda\,\mathrm{e}^{-\lambda x}$	$\mathbb{R}_+$	$\lambda>0$
Beta	$\mathsf{Beta}(\alpha,\beta)$	$\dfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\,x^{\alpha-1}(1-x)^{\beta-1}$	$[0,1]$	$\alpha,\beta>0$
Weibull	$\mathsf{Weib}(\alpha,\lambda)$	$\alpha\lambda\,(\lambda x)^{\alpha-1}\,\mathrm{e}^{-(\lambda x)^\alpha}$	$\mathbb{R}_+$	$\alpha,\lambda>0$
Pareto	$\mathsf{Pareto}(\alpha,\lambda)$	$\alpha\lambda\,(1+\lambda x)^{-(\alpha+1)}$	$\mathbb{R}_+$	$\alpha,\lambda>0$

Table 1.2 Commonly used discrete distributions.

Name	Notation	$f(x)$	$x \in$	Params.
Bernoulli	$\mathsf{Ber}(p)$	$p^x(1-p)^{1-x}$	$\{0,1\}$	$0\leqslant p\leqslant 1$
Binomial	$\mathsf{Bin}(n,p)$	$\binom{n}{x}p^x(1-p)^{n-x}$	$\{0,1,\ldots,n\}$	$0\leqslant p\leqslant 1$, $n\in\mathbb{N}$
Discrete uniform	$\mathsf{DU}\{1,\ldots,n\}$	$\dfrac{1}{n}$	$\{1,\ldots,n\}$	$n\in\{1,2,\ldots\}$
Geometric	$\mathsf{G}(p)$	$p(1-p)^{x-1}$	$\{1,2,\ldots\}$	$0\leqslant p\leqslant 1$
Poisson	$\mathsf{Poi}(\lambda)$	$\mathrm{e}^{-\lambda}\dfrac{\lambda^x}{x!}$	$\mathbb{N}$	$\lambda>0$

1.5 EXPECTATION

It is often useful to consider various numerical characteristics of a random variable. One such quantity is the expectation, which measures the mean value of the distribution.

Definition 1.5.1 (Expectation) Let X be a random variable with pdf f. The *expectation* (or expected value or mean) of X, denoted by $\mathbb{E}[X]$ (or sometimes μ), is defined by

$$\mathbb{E}[X] = \begin{cases} \sum_x x\, f(x) & \text{discrete case,} \\ \int_{-\infty}^{\infty} x\, f(x)\, dx & \text{continuous case.} \end{cases}$$

If X is a random variable, then a function of X, such as X^2 or $\sin(X)$, is again a random variable. Moreover, the expected value of a function of X is simply a weighted average of the possible values that this function can take. That is, for any real function h

$$\mathbb{E}[h(X)] = \begin{cases} \sum_x h(x)\, f(x) & \text{discrete case,} \\ \int_{-\infty}^{\infty} h(x)\, f(x)\, dx & \text{continuous case.} \end{cases}$$

Another useful quantity is the variance, which measures the spread or dispersion of the distribution.

Definition 1.5.2 (Variance) The *variance* of a random variable X, denoted by $\mathrm{Var}(X)$ (or sometimes σ^2), is defined by

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 \,.$$

The square root of the variance is called the *standard deviation*. Table 1.3 lists the expectations and variances for some well-known distributions.

Table 1.3 Expectations and variances for some well-known distributions.

Dist.	$\mathbb{E}[X]$	$\mathrm{Var}(X)$	Dist.	$\mathbb{E}[X]$	$\mathrm{Var}(X)$
$\mathsf{Bin}(n,p)$	np	$np(1-p)$	$\mathsf{Gamma}(\alpha,\lambda)$	$\frac{\alpha}{\lambda}$	$\frac{\alpha}{\lambda^2}$
$\mathsf{G}(p)$	$\frac{1}{p}$	$\frac{1-p}{p^2}$	$\mathsf{N}(\mu,\sigma^2)$	μ	σ^2
$\mathsf{Poi}(\lambda)$	λ	λ	$\mathsf{Beta}(\alpha,\beta)$	$\frac{\alpha}{\alpha+\beta}$	$\frac{\alpha\beta}{(\alpha+\beta)^2(1+\alpha+\beta)}$
$\mathsf{U}(\alpha,\beta)$	$\frac{\alpha+\beta}{2}$	$\frac{(\beta-\alpha)^2}{12}$	$\mathsf{Weib}(\alpha,\lambda)$	$\frac{\Gamma(1/\alpha)}{\alpha\lambda}$	$\frac{2\Gamma(2/\alpha)}{\alpha} - \left(\frac{\Gamma(1/\alpha)}{\alpha\lambda}\right)^2$
$\mathsf{Exp}(\lambda)$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$			

The mean and the variance do not give, in general, enough information to completely specify the distribution of a random variable. However, they may provide useful bounds.

We discuss two such bounds. Suppose X can only take nonnegative values and has pdf f. For any $x > 0$, we can write

$$\begin{aligned}\mathbb{E}[X] &= \int_0^x t f(t)\,dt + \int_x^\infty t f(t)\,dt \geqslant \int_x^\infty t f(t)\,dt \\ &\geqslant \int_x^\infty x f(t)\,dt = x\,\mathbb{P}(X \geqslant x)\,,\end{aligned}$$

from which follows the *Markov inequality:* if $X \geqslant 0$, then for all $x > 0$,

$$\mathbb{P}(X \geqslant x) \leqslant \frac{\mathbb{E}[X]}{x}\,. \tag{1.9}$$

If we also know the variance of a random variable, we can give a tighter bound. Namely, for any random variable X with mean μ and variance σ^2, we have

$$\mathbb{P}(|X-\mu| \geqslant x) \leqslant \frac{\sigma^2}{x^2}\,. \tag{1.10}$$

This is called the *Chebyshev inequality*. The proof is as follows: Let $D^2 = (X-\mu)^2$; then, by the Markov inequality (1.9) and the definition of the variance,

$$\mathbb{P}(D^2 \geqslant x^2) \leqslant \frac{\sigma^2}{x^2}\,.$$

Also, note that the event $\{D^2 \geqslant x^2\}$ is equivalent to the event $\{|X-\mu| \geqslant x\}$, so that (1.10) follows.

1.6 JOINT DISTRIBUTIONS

Often a random experiment is described by more than one random variable. The theory for multiple random variables is similar to that for a single random variable.

Let $X_1, \ldots, X_n$ be random variables describing some random experiment. We can accumulate these into a *random vector* $\mathbf{X} = (X_1, \ldots, X_n)$. More generally, a collection $\{X_t, t \in \mathscr{T}\}$ of random variables is called a *stochastic process*. The set $\mathscr{T}$ is called the *parameter set* or *index set* of the process. It may be discrete (such as $\mathbb{N}$ or $\{1, \ldots, 10\}$) or continuous (for example, $\mathbb{R}_+ = [0, \infty)$ or $[1, 10]$). The set of possible values for the stochastic process is called the *state space*.

The joint distribution of $X_1, \ldots, X_n$ is specified by the *joint cdf*

$$F(x_1, \ldots, x_n) = \mathbb{P}(X_1 \leqslant x_1, \ldots, X_n \leqslant x_n)\,.$$

The *joint pdf* f is given, in the discrete case, by $f(x_1, \ldots, x_n) = \mathbb{P}(X_1 = x_1, \ldots, X_n = x_n)$, and in the continuous case f is such that

$$\mathbb{P}(\mathbf{X} \in \mathscr{B}) = \int_{\mathscr{B}} f(x_1, \ldots, x_n)\; dx_1 \ldots dx_n$$

for any (measurable) region $\mathscr{B}$ in $\mathbb{R}^n$. The marginal pdfs can be recovered from the joint pdf by integration or summation. For example, in the case of a continuous random vector (X, Y) with joint pdf f, the pdf f_X of X is found as

$$f_X(x) = \int f(x, y)\,dy\,.$$

Suppose X and Y are both discrete or both continuous, with joint pdf f, and suppose $f_X(x) > 0$. Then the *conditional pdf* of Y given $X = x$ is given by

$$f_{Y|X}(y \mid x) = \frac{f(x,y)}{f_X(x)} \quad \text{for all } y \,.$$

The corresponding *conditional expectation* is (in the continuous case)

$$\mathbb{E}[Y \mid X = x] = \int y \, f_{Y|X}(y \mid x) \, dy \,.$$

Note that $\mathbb{E}[Y \mid X = x]$ is a function of x, say $h(x)$. The corresponding random variable $h(X)$ is written as $\mathbb{E}[Y \mid X]$. It can be shown (see, for example, [4]) that its expectation is simply the expectation of Y, that is,

$$\mathbb{E}[\mathbb{E}[Y \mid X]] = \mathbb{E}[Y] \,. \tag{1.11}$$

When the conditional distribution of Y given X is identical to that of Y, X and Y are said to be independent. More precisely:

Definition 1.6.1 (Independent Random Variables) The random variables $X_1, \ldots, X_n$ are called *independent* if for all events $\{X_i \in A_i\}$ with $A_i \subset \mathbb{R}$, $i = 1, \ldots, n$

$$\mathbb{P}(X_1 \in A_1, \ldots, X_n \in A_n) = \mathbb{P}(X_1 \in A_1) \cdots \mathbb{P}(X_n \in A_n) \,.$$

A direct consequence of the above definition for independence is that random variables $X_1, \ldots, X_n$ with joint pdf f (discrete or continuous) are independent if and only if

$$f(x_1, \ldots, x_n) = f_{X_1}(x_1) \cdots f_{X_n}(x_n) \tag{1.12}$$

for all $x_1, \ldots, x_n$, where $\{f_{X_i}\}$ are the marginal pdfs.

■ **EXAMPLE 1.4 Bernoulli Sequence**

Consider the experiment where we flip a biased coin n times, with probability p of heads. We can model this experiment in the following way. For $i = 1, \ldots, n$ let X_i be the result of the i-th toss: $\{X_i = 1\}$ means heads (or success), $\{X_i = 0\}$ means tails (or failure). Also, let

$$\mathbb{P}(X_i = 1) = p = 1 - \mathbb{P}(X_i = 0), \quad i = 1, 2, \ldots, n \,.$$

Finally, assume that $X_1, \ldots, X_n$ are independent. The sequence $\{X_i, i = 1, 2, \ldots\}$ is called a *Bernoulli sequence* or *Bernoulli process* with success probability p. Let $X = X_1 + \cdots + X_n$ be the total number of successes in n trials (tosses of the coin). Denote by $\mathscr{B}$ the set of all binary vectors $\mathbf{x} = (x_1, \ldots, x_n)$ such that $\sum_{i=1}^n x_i = k$. Note that $\mathscr{B}$ has $\binom{n}{k}$ elements. We now have

$$\begin{aligned}
\mathbb{P}(X = k) &= \sum_{\mathbf{x} \in \mathscr{B}} \mathbb{P}(X_1 = x_1, \ldots, X_n = x_n) \\
&= \sum_{\mathbf{x} \in \mathscr{B}} \mathbb{P}(X_1 = x_1) \cdots \mathbb{P}(X_n = x_n) = \sum_{\mathbf{x} \in \mathscr{B}} p^k (1-p)^{n-k} \\
&= \binom{n}{k} p^k (1-p)^{n-k} \,.
\end{aligned}$$

In other words, $X \sim \mathsf{Bin}(n, p)$. Compare this with Example 1.2.

Remark 1.6.1 An *infinite* sequence $X_1, X_2, \ldots$ of random variables is called independent if for any finite choice of parameters $i_1, i_2, \ldots, i_n$ (none of them the same) the random variables $X_{i_1}, \ldots, X_{i_n}$ are independent. Many probabilistic models involve random variables $X_1, X_2, \ldots$ that are *independent and identically distributed*, abbreviated as *iid*. We will use this abbreviation throughout this book.

Similar to the one-dimensional case, the expected value of any real-valued function h of $X_1, \ldots, X_n$ is a weighted average of all values that this function can take. Specifically, in the continuous case,

$$\mathbb{E}[h(X_1, \ldots, X_n)] = \int \cdots \int h(x_1, \ldots, x_n)\, f(x_1, \ldots, x_n)\, dx_1 \ldots dx_n \,.$$

As a direct consequence of the definitions of expectation and independence, we have

$$\mathbb{E}[a + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n] = a + b_1 \mu_1 + \cdots + b_n \mu_n \tag{1.13}$$

for any sequence of random variables $X_1, X_2, \ldots, X_n$ with expectations $\mu_1, \mu_2, \ldots, \mu_n$, where $a, b_1, b_2, \ldots, b_n$ are constants. Similarly, for *independent* random variables one has

$$\mathbb{E}[X_1 X_2 \cdots X_n] = \mu_1\, \mu_2 \cdots \mu_n \,.$$

The *covariance* of two random variables X and Y with expectations $\mathbb{E}[X] = \mu_X$ and $\mathbb{E}[Y] = \mu_Y$, respectively, is defined as

$$\mathrm{Cov}(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)] \,.$$

This is a measure for the amount of linear dependency between the variables. A scaled version of the covariance is given by the *correlation coefficient*,

$$\varrho(X, Y) = \frac{\mathrm{Cov}(X, Y)}{\sigma_X\, \sigma_Y},$$

where $\sigma_X^2 = \mathrm{Var}(X)$ and $\sigma_Y^2 = \mathrm{Var}(Y)$. It can be shown that the correlation coefficient always lies between -1 and 1; see Problem 1.13.

For easy reference, Table 1.4 lists some important properties of the variance and covariance. The proofs follow directly from the definitions of covariance and variance and the properties of the expectation.

Table 1.4 Properties of variance and covariance.

1	$\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$
2	$\mathrm{Var}(aX + b) = a^2 \mathrm{Var}(X)$
3	$\mathrm{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]$
4	$\mathrm{Cov}(X, Y) = \mathrm{Cov}(Y, X)$
5	$\mathrm{Cov}(aX + bY, Z) = a\,\mathrm{Cov}(X, Z) + b\,\mathrm{Cov}(Y, Z)$
6	$\mathrm{Cov}(X, X) = \mathrm{Var}(X)$
7	$\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) + 2\,\mathrm{Cov}(X, Y)$
8	X and Y indep. $\Longrightarrow \mathrm{Cov}(X, Y) = 0$

As a consequence of properties 2 and 7, for any sequence of *independent* random variables $X_1, \ldots, X_n$ with variances $\sigma_1^2, \ldots, \sigma_n^2$

$$\operatorname{Var}(a + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n) = b_1^2 \sigma_1^2 + \cdots + b_n^2 \sigma_n^2 \tag{1.14}$$

for any choice of constants a and $b_1, \ldots, b_n$.

For random vectors, such as $\mathbf{X} = (X_1, \ldots, X_n)^T$, it is convenient to write the expectations and covariances in vector notation.

Definition 1.6.2 (Expectation Vector and Covariance Matrix) For any random vector $\mathbf{X}$ we define the *expectation vector* as the vector of expectations

$$\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^T = (\mathbb{E}[X_1], \ldots, \mathbb{E}[X_n])^T .$$

The *covariance matrix* Σ is defined as the matrix whose (i, j)-th element is

$$\operatorname{Cov}(X_i, X_j) = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)] .$$

If we define the expectation of a vector (matrix) to be the vector (matrix) of expectations, then we can write

$$\boldsymbol{\mu} = \mathbb{E}[\mathbf{X}]$$

and

$$\Sigma = \mathbb{E}[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^T] .$$

Note that $\boldsymbol{\mu}$ and Σ take on the same role as μ and σ^2 in the one-dimensional case.

Remark 1.6.2 Note that any covariance matrix Σ is *symmetric*. In fact (see Problem 1.16), it is *positive semidefinite*, that is, for any (column) vector $\mathbf{u}$,

$$\mathbf{u}^T \Sigma \mathbf{u} \geqslant 0 .$$

1.7 FUNCTIONS OF RANDOM VARIABLES

Suppose $X_1, \ldots, X_n$ are measurements of a random experiment. Often we are only interested in certain *functions* of the measurements rather than the individual measurements themselves. We give a number of examples.

■ **EXAMPLE 1.5**

Let X be a continuous random variable with pdf f_X and let $Z = aX + b$, where $a \neq 0$. We wish to determine the pdf f_Z of Z. Suppose that $a > 0$. We have for any z

$$F_Z(z) = \mathbb{P}(Z \leqslant z) = \mathbb{P}\big(X \leqslant (z - b)/a\big) = F_X\big((z - b)/a\big) .$$

Differentiating this with respect to z gives $f_Z(z) = f_X\big((z - b)/a\big)/a$. For $a < 0$ we similarly obtain $f_Z(z) = f_X\big((z - b)/a\big)/(-a)$. Thus, in general,

$$f_Z(z) = \frac{1}{|a|} f_X\left(\frac{z - b}{a}\right) . \tag{1.15}$$

■ **EXAMPLE 1.6**

Generalizing the previous example, suppose that $Z = g(X)$ for some monotonically increasing function g. To find the pdf of Z from that of X we first write

$$F_Z(z) = \mathbb{P}(Z \leqslant z) = \mathbb{P}\left(X \leqslant g^{-1}(z)\right) = F_X\left(g^{-1}(z)\right) ,$$

where g^{-1} is the inverse of g. Differentiating with respect to z now gives

$$f_Z(z) = f_X(g^{-1}(z))\,\frac{d}{dz}g^{-1}(z) = \frac{f_X(g^{-1}(z))}{g'(g^{-1}(z))} . \tag{1.16}$$

For monotonically decreasing functions, $\frac{d}{dz}g^{-1}(z)$ in the first equation needs to be replaced with its negative value.

■ **EXAMPLE 1.7 Order Statistics**

Let $X_1, \ldots, X_n$ be an iid sequence of random variables with common pdf f and cdf F. In many applications one is interested in the distribution of the *order statistics* $X_{(1)}, X_{(2)}, \ldots, X_{(n)}$, where $X_{(1)}$ is the smallest of the $\{X_i, i = 1, \ldots, n\}$, $X_{(2)}$ is the second smallest, and so on. The cdf of $X_{(n)}$ follows from

$$\mathbb{P}(X_{(n)} \leqslant x) = \mathbb{P}(X_1 \leqslant x, \ldots, X_n \leqslant x) = \prod_{i=1}^{n} \mathbb{P}(X_i \leqslant x) = (F(x))^n .$$

Similarly,

$$\mathbb{P}(X_{(1)} > x) = \mathbb{P}(X_1 > x, \ldots, X_n > x) = \prod_{i=1}^{n} \mathbb{P}(X_i > x) = (1 - F(x))^n .$$

Moreover, because all orderings of $X_1, \ldots, X_n$ are equally likely, it follows that the joint pdf of the ordered sample is, on the wedge $\{(x_1, \ldots, x_n) : x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n\}$, simply $n!$ times the joint density of the unordered sample and zero elsewhere.

1.7.1 Linear Transformations

Let $\mathbf{x} = (x_1, \ldots, x_n)^T$ be a column vector in $\mathbb{R}^n$ and A an $m \times n$ matrix. The mapping $\mathbf{x} \mapsto \mathbf{z}$, with $\mathbf{z} = A\mathbf{x}$, is called a *linear transformation*. Now consider a *random* vector $\mathbf{X} = (X_1, \ldots, X_n)^T$, and let

$$\mathbf{Z} = A\mathbf{X} .$$

Then $\mathbf{Z}$ is a random vector in $\mathbb{R}^m$. In principle, if we know the joint distribution of $\mathbf{X}$, then we can derive the joint distribution of $\mathbf{Z}$. Let us first see how the expectation vector and covariance matrix are transformed.

Theorem 1.7.1 *If $\mathbf{X}$ has an expectation vector $\boldsymbol{\mu}_{\mathbf{X}}$ and covariance matrix $\Sigma_{\mathbf{X}}$, then the expectation vector and covariance matrix of $\mathbf{Z} = A\mathbf{X}$ are given by*

$$\boldsymbol{\mu}_{\mathbf{Z}} = A\boldsymbol{\mu}_{\mathbf{X}} \tag{1.17}$$

and

$$\Sigma_{\mathbf{Z}} = A\,\Sigma_{\mathbf{X}}\,A^T . \tag{1.18}$$

Proof: We have $\mu_{\mathbf{Z}} = \mathbb{E}[\mathbf{Z}] = \mathbb{E}[A\mathbf{X}] = A\,\mathbb{E}[\mathbf{X}] = A\mu_{\mathbf{X}}$ and

$$\begin{aligned}\Sigma_{\mathbf{Z}} &= \mathbb{E}[(\mathbf{Z}-\mu_{\mathbf{Z}})(\mathbf{Z}-\mu_{\mathbf{Z}})^T] = \mathbb{E}[A(\mathbf{X}-\mu_{\mathbf{X}})(A(\mathbf{X}-\mu_{\mathbf{X}}))^T] \\ &= A\,\mathbb{E}[(\mathbf{X}-\mu_{\mathbf{X}})(\mathbf{X}-\mu_{\mathbf{X}})^T]A^T \\ &= A\,\Sigma_{\mathbf{X}}\,A^T\,.\end{aligned}$$

□

Suppose that A is an invertible $n \times n$ matrix. If $\mathbf{X}$ has a joint density $f_{\mathbf{X}}$, what is the joint density $f_{\mathbf{Z}}$ of $\mathbf{Z}$? Consider Figure 1.1. For any fixed $\mathbf{x}$, let $\mathbf{z} = A\mathbf{x}$. Hence, $\mathbf{x} = A^{-1}\mathbf{z}$. Consider the n-dimensional cube $C = [z_1, z_1 + h] \times \cdots \times [z_n, z_n + h]$. Let D be the image of C under A^{-1}, that is, the parallelepiped of all points $\mathbf{x}$ such that $A\mathbf{x} \in C$. Then,

$$\mathbb{P}(\mathbf{Z} \in C) \approx h^n\, f_{\mathbf{Z}}(\mathbf{z})\,.$$

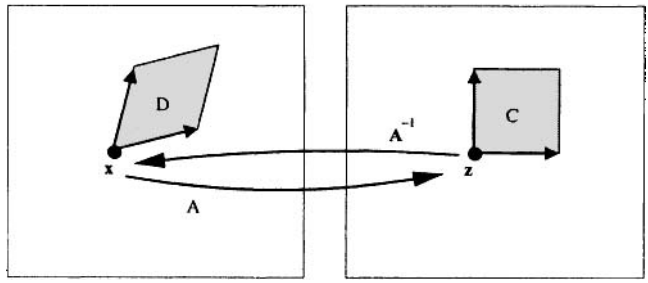


Figure 1.1 Linear transformation.

Now recall from linear algebra (see, for example, [6]) that any matrix B linearly transforms an n-dimensional rectangle with volume V into an n-dimensional parallelepiped with volume $V\,|B|$, where $|B| = |\det(B)|$. Thus,

$$\mathbb{P}(\mathbf{Z} \in C) = \mathbb{P}(\mathbf{X} \in D) \approx h^n |A^{-1}|\, f_{\mathbf{X}}(\mathbf{x}) = h^n |A|^{-1}\, f_{\mathbf{X}}(\mathbf{x})\,.$$

Letting h go to 0, we obtain

$$f_{\mathbf{Z}}(\mathbf{z}) = \frac{f_{\mathbf{X}}(A^{-1}\mathbf{z})}{|A|}, \quad \mathbf{z} \in \mathbb{R}^n. \tag{1.19}$$

1.7.2 General Transformations

We can apply reasoning similar to that above to deal with general transformations $\mathbf{x} \mapsto g(\mathbf{x})$, written out:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \mapsto \begin{pmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_n(\mathbf{x}) \end{pmatrix}.$$

For a fixed $\mathbf{x}$, let $\mathbf{z} = g(\mathbf{x})$. Suppose g is invertible; hence, $\mathbf{x} = g^{-1}(\mathbf{z})$. Any infinitesimal n-dimensional rectangle at $\mathbf{x}$ with volume V is transformed into an n-dimensional

parallelepiped at $\mathbf{z}$ with volume $V\,|J_{\mathbf{x}}(\boldsymbol{g})|$, where $J_{\mathbf{x}}(\boldsymbol{g})$ is the *matrix of Jacobi* at $\mathbf{x}$ of the transformation $\boldsymbol{g}$, that is,

$$J_{\mathbf{x}}(\boldsymbol{g}) = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_n} \end{pmatrix} .$$

Now consider a random column vector $\mathbf{Z} = \boldsymbol{g}(\mathbf{X})$. Let C be a small cube around $\mathbf{z}$ with volume h^n. Let D be the image of C under $\boldsymbol{g}^{-1}$. Then, as in the linear case,

$$\mathbb{P}(\mathbf{Z} \in C) \approx h^n f_{\mathbf{Z}}(\mathbf{z}) \approx h^n |J_{\mathbf{z}}(\boldsymbol{g}^{-1})|\, f_{\mathbf{X}}(\mathbf{x}) .$$

Hence, we have the transformation rule

$$f_{\mathbf{Z}}(\mathbf{z}) = f_{\mathbf{X}}(\boldsymbol{g}^{-1}(\mathbf{z}))\, |J_{\mathbf{z}}(\boldsymbol{g}^{-1})|, \quad \mathbf{z} \in \mathbb{R}^n. \tag{1.20}$$

(Note: $|J_{\mathbf{z}}(\boldsymbol{g}^{-1})| = 1/|J_{\mathbf{x}}(\boldsymbol{g})|$.)

Remark 1.7.1 In most coordinate transformations it is $\boldsymbol{g}^{-1}$ that is given — that is, an expression for $\mathbf{x}$ as a function of $\mathbf{z}$ rather than $\boldsymbol{g}$.

1.8 TRANSFORMS

Many calculations and manipulations involving probability distributions are facilitated by the use of transforms. Two typical examples are the *probability generating function* of a positive integer-valued random variable N, defined by

$$G(z) = \mathbb{E}[z^N] = \sum_{k=0}^{\infty} z^k\, \mathbb{P}(N = k) , \quad |z| \leqslant 1 ,$$

and the *Laplace transform* of a positive random variable X defined, for $s \geqslant 0$, by

$$L(s) = \mathbb{E}[\mathrm{e}^{-sX}] = \begin{cases} \sum_x \mathrm{e}^{-sx} f(x) & \text{discrete case,} \\ \int_0^{\infty} \mathrm{e}^{-sx} f(x)\, dx & \text{continuous case.} \end{cases}$$

All transforms share an important *uniqueness property*: two distributions are the same if and only if their respective transforms are the same.

■ **EXAMPLE 1.8**

Let $M \sim \mathsf{Poi}(\mu)$; then its probability generating function is given by

$$G(z) = \sum_{k=0}^{\infty} z^k\, \mathrm{e}^{-\mu}\, \frac{\mu^k}{k!} = \mathrm{e}^{-\mu} \sum_{k=0}^{\infty} \frac{(z\mu)^k}{k!} = \mathrm{e}^{-\mu} \mathrm{e}^{z\mu} = \mathrm{e}^{-\mu(1-z)} . \tag{1.21}$$

Now let $N \sim \mathsf{Poi}(\nu)$ independently of M. Then the probability generating function of $M + N$ is given by

$$\mathbb{E}[z^{M+N}] = \mathbb{E}[z^M]\, \mathbb{E}[z^N] = \mathrm{e}^{-\mu(1-z)} \mathrm{e}^{-\nu(1-z)} = \mathrm{e}^{-(\mu+\nu)(1-z)} .$$

Thus, by the uniqueness property, $M + N \sim \mathsf{Poi}(\mu + \nu)$.

■ **EXAMPLE 1.9**

The Laplace transform of $X \sim \mathsf{Gamma}(\alpha, \lambda)$ is given by

$$\begin{aligned}\mathbb{E}[\mathrm{e}^{-sX}] &= \int_0^\infty \frac{\mathrm{e}^{-\lambda x}\,\lambda^\alpha\, x^{\alpha-1}}{\Gamma(\alpha)}\,\mathrm{e}^{-sx}\,dx \\ &= \left(\frac{\lambda}{\lambda+s}\right)^\alpha \int_0^\infty \frac{\mathrm{e}^{-(\lambda+s)x}\,(\lambda+s)^\alpha\, x^{\alpha-1}}{\Gamma(\alpha)}\,dx \\ &= \left(\frac{\lambda}{\lambda+s}\right)^\alpha .\end{aligned}$$

As a special case, the Laplace transform of the $\mathsf{Exp}(\lambda)$ distribution is given by $\lambda/(\lambda+s)$. Now let $X_1, \ldots, X_n$ be iid $\mathsf{Exp}(\lambda)$ random variables. The Laplace transform of $S_n = X_1 + \cdots + X_n$ is

$$\mathbb{E}[\mathrm{e}^{-sS_n}] = \mathbb{E}[\mathrm{e}^{-sX_1}\cdots\mathrm{e}^{-sX_n}] = \mathbb{E}[\mathrm{e}^{-sX_1}]\cdots\mathbb{E}[\mathrm{e}^{-sX_n}] = \left(\frac{\lambda}{\lambda+s}\right)^n ,$$

which shows that $S_n \sim \mathsf{Gamma}(n, \lambda)$.

1.9 JOINTLY NORMAL RANDOM VARIABLES

It is helpful to view normally distributed random variables as simple transformations of *standard normal* — that is, $\mathsf{N}(0,1)$-distributed — random variables. In particular, let $X \sim \mathsf{N}(0,1)$. Then, X has density f_X given by

$$f_X(x) = \frac{1}{\sqrt{2\pi}}\,\mathrm{e}^{-\frac{x^2}{2}}.$$

Now consider the transformation $Z = \mu + \sigma X$. Then, by (1.15), Z has density

$$f_Z(z) = \frac{1}{\sqrt{2\pi\sigma^2}}\,\mathrm{e}^{-\frac{(z-\mu)^2}{2\sigma^2}}.$$

In other words, $Z \sim \mathsf{N}(\mu, \sigma^2)$. We can also state this as follows: if $Z \sim \mathsf{N}(\mu, \sigma^2)$, then $(Z-\mu)/\sigma \sim \mathsf{N}(0,1)$. This procedure is called *standardization*.

We now generalize this to n dimensions. Let $X_1, \ldots, X_n$ be independent and standard normal random variables. The joint pdf of $\mathbf{X} = (X_1, \ldots, X_n)^T$ is given by

$$f_{\mathbf{X}}(\mathbf{x}) = (2\pi)^{-n/2}\mathrm{e}^{-\frac{1}{2}\mathbf{x}^T\mathbf{x}}, \quad \mathbf{x} \in \mathbb{R}^n. \tag{1.22}$$

Consider the *affine* transformation (that is, a linear transformation plus a constant vector)

$$\mathbf{Z} = \boldsymbol{\mu} + B\,\mathbf{X} \tag{1.23}$$

for some $m \times n$ matrix B. Note that, by Theorem 1.7.1, $\mathbf{Z}$ has expectation vector $\boldsymbol{\mu}$ and covariance matrix $\Sigma = BB^T$. Any random vector of the form (1.23) is said to have a *jointly normal* or *multivariate normal* distribution. We write $\mathbf{Z} \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma)$. Suppose B is an invertible $n \times n$ matrix. Then, by (1.19), the density of $\mathbf{Y} = \mathbf{Z} - \boldsymbol{\mu}$ is given by

$$f_{\mathbf{Y}}(\mathbf{y}) = \frac{1}{|B|\sqrt{(2\pi)^n}}\,\mathrm{e}^{-\frac{1}{2}(B^{-1}\mathbf{y})^T B^{-1}\mathbf{y}} = \frac{1}{|B|\sqrt{(2\pi)^n}}\,\mathrm{e}^{-\frac{1}{2}\mathbf{y}^T(B^{-1})^T B^{-1}\mathbf{y}}.$$

We have $|B| = \sqrt{|\Sigma|}$ and $(B^{-1})^T B^{-1} = (B^T)^{-1} B^{-1} = (BB^T)^{-1} = \Sigma^{-1}$, so that

$$f_{\mathbf{Y}}(\mathbf{y}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \mathrm{e}^{-\frac{1}{2}\,\mathbf{y}^T \Sigma^{-1} \mathbf{y}} .$$

Because $\mathbf{Z}$ is obtained from $\mathbf{Y}$ by simply adding a constant vector $\boldsymbol{\mu}$, we have $f_{\mathbf{Z}}(\mathbf{z}) = f_{\mathbf{Y}}(\mathbf{z} - \boldsymbol{\mu})$ and therefore

$$f_{\mathbf{Z}}(\mathbf{z}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \mathrm{e}^{-\frac{1}{2}\,(\mathbf{z}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{z}-\boldsymbol{\mu})}, \quad \mathbf{z} \in \mathbb{R}^n . \tag{1.24}$$

Note that this formula is very similar to the one-dimensional case.

Conversely, given a covariance matrix $\Sigma = (\sigma_{ij})$, there exists a unique lower triangular matrix

$$B = \begin{pmatrix} b_{11} & 0 & \cdots & 0 \\ b_{21} & b_{22} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix} \tag{1.25}$$

such that $\Sigma = BB^T$. This matrix can be obtained efficiently via the *Cholesky square root method*, see Section A.1 of the Appendix.

1.10 LIMIT THEOREMS

We briefly discuss two of the main results in probability: the law of large numbers and the central limit theorem. Both are associated with sums of independent random variables.

Let $X_1, X_2, \ldots$ be iid random variables with expectation μ and variance σ^2. For each n let $S_n = X_1 + \cdots + X_n$. Since $X_1, X_2, \ldots$ are iid, we have $\mathbb{E}[S_n] = n\,\mathbb{E}[X_1] = n\mu$ and $\mathrm{Var}(S_n) = n\,\mathrm{Var}(X_1) = n\sigma^2$.

The law of large numbers states that S_n/n is close to μ for large n. Here is the more precise statement.

Theorem 1.10.1 (Strong Law of Large Numbers) *If $X_1, \ldots, X_n$ are iid with expectation μ, then*

$$\mathbb{P}\left(\lim_{n\to\infty} \frac{S_n}{n} = \mu\right) = 1 .$$

The central limit theorem describes the limiting distribution of S_n (or S_n/n), and it applies to both continuous and discrete random variables. Loosely, it states that the random sum S_n has a distribution that is approximately normal, when n is large. The more precise statement is given next.

Theorem 1.10.2 (Central Limit Theorem) *If $X_1, \ldots, X_n$ are iid with expectation μ and variance $\sigma^2 < \infty$, then for all $x \in \mathbb{R}$,*

$$\lim_{n\to\infty} \mathbb{P}\left(\frac{S_n - n\mu}{\sigma\sqrt{n}} \leqslant x\right) = \Phi(x) ,$$

where Φ is the cdf of the standard normal distribution.

In other words, S_n has a distribution that is approximately normal, with expectation $n\mu$ and variance $n\sigma^2$. To see the central limit theorem in action, consider Figure 1.2. The left part shows the pdfs of $S_1, \ldots, S_4$ for the case where the $\{X_i\}$ have a U[0, 1] distribution. The right part shows the same for the Exp(1) distribution. We clearly see convergence to a bell-shaped curve, characteristic of the normal distribution.

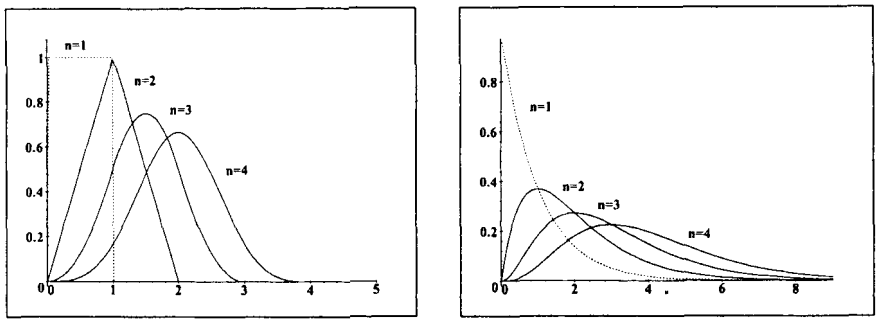


Figure 1.2 Illustration of the central limit theorem for (left) the uniform distribution and (right) the exponential distribution.

A direct consequence of the central limit theorem and the fact that a $\mathsf{Bin}(n, p)$ random variable X can be viewed as the sum of n iid $\mathsf{Ber}(p)$ random variables, $X = X_1 + \cdots + X_n$, is that for large n

$$\mathbb{P}(X \leqslant k) \approx \mathbb{P}(Y \leqslant k)\,, \tag{1.26}$$

with $Y \sim \mathsf{N}(np, np(1-p))$. As a rule of thumb, this *normal approximation to the binomial distribution* is accurate if both np and $n(1-p)$ are larger than 5.

There is also a central limit theorem for random vectors. The multidimensional version is as follows: Let $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be iid random vectors with expectation vector $\boldsymbol{\mu}$ and covariance matrix Σ. Then for large n the random vector $\mathbf{X}_1 + \cdots + \mathbf{X}_n$ has approximately a multivariate normal distribution with expectation vector $n\boldsymbol{\mu}$ and covariance matrix $n\Sigma$.

1.11 POISSON PROCESSES

The Poisson process is used to model certain kinds of arrivals or patterns. Imagine, for example, a telescope that can detect individual photons from a faraway galaxy. The photons arrive at random times $T_1, T_2, \ldots$. Let N_t denote the number of arrivals in the time interval $[0, t]$, that is, $N_t = \sup\{k : T_k \leqslant t\}$. Note that the number of arrivals in an interval $I = (a, b]$ is given by $N_b - N_a$. We will also denote it by $N(a, b]$. A sample path of the arrival counting process $\{N_t, t \geqslant 0\}$ is given in Figure 1.3.

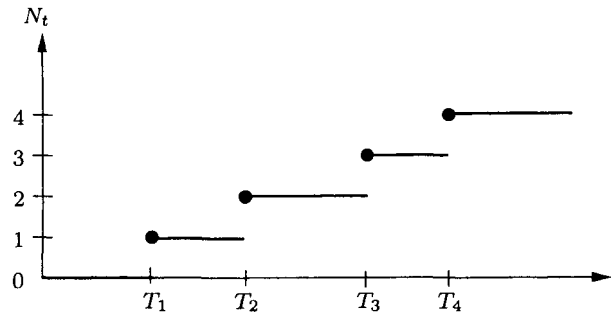


Figure 1.3 A sample path of the arrival counting process $\{N_t, t \geqslant 0\}$.

For this particular arrival process, one would assume that the number of arrivals in an interval (a, b) is independent of the number of arrivals in interval (c, d) when the two intervals do not intersect. Such considerations lead to the following definition:

Definition 1.11.1 (Poisson Process) An arrival counting process $N = \{N_t\}$ is called a *Poisson process* with *rate* $\lambda > 0$ if

(a) The numbers of points in nonoverlapping intervals are independent.

(b) The number of points in interval I has a Poisson distribution with mean $\lambda \times \text{length}(I)$.

Combining (a) and (b) we see that the number of arrivals in any small interval $(t, t+h]$ is independent of the arrival process up to time t and has a $\mathsf{Poi}(\lambda h)$ distribution. In particular, the conditional probability that exactly one arrival occurs during the time interval $(t, t+h]$ is $\mathbb{P}(N(t, t+h] = 1 \mid N_t) = \mathrm{e}^{-\lambda h}\, \lambda h \approx \lambda h$. Similarly, the probability of no arrivals is approximately $1 - \lambda h$ for small h. In other words, λ is the *rate* at which arrivals occur. Notice also that since $N_t \sim \mathsf{Poi}(\lambda t)$, the expected number of arrivals in $[0, t]$ is λt, that is, $\mathbb{E}[N_t] = \lambda t$. In Definition 1.11.1 N is seen as a random counting measure, where $N(I)$ counts the random number of arrivals in set I.

An important relationship between N_t and T_n is

$$\{N_t \geqslant n\} = \{T_n \leqslant t\}\,. \tag{1.27}$$

In other words, the number of arrivals in $[0, t]$ is at least n if and only if the n-th arrival occurs at or before time t. As a consequence, we have

$$\begin{aligned} \mathbb{P}(T_n \leqslant t) &= \mathbb{P}(N_t \geqslant n) = 1 - \sum_{k=0}^{n-1} \mathbb{P}(N_t = k) \\ &= 1 - \sum_{k=0}^{n-1} \frac{\mathrm{e}^{-\lambda t}(\lambda t)^k}{k!}\,, \end{aligned}$$

which corresponds exactly to the cdf of the $\mathsf{Gamma}(n, \lambda)$ distribution; see Problem 1.17. Thus,

$$T_n \sim \mathsf{Gamma}(n, \lambda). \tag{1.28}$$

Hence, each T_n has the same distribution as the sum of n independent $\mathsf{Exp}(\lambda)$-distributed random variables. This corresponds with the second important characterization of a Poisson process:

> *An arrival counting process $\{N_t\}$ is a Poisson process with rate λ if and only if the interarrival times $A_1 = T_1, A_2 = T_2 - T_1, \ldots$ are independent and $\mathsf{Exp}(\lambda)$-distributed random variables.*

Poisson and Bernoulli processes are akin, and much can be learned about Poisson processes via the following *Bernoulli approximation*. Let $N = \{N_t\}$ be a Poisson process with parameter λ. We divide the time axis into small time intervals $[0, h), [h, 2h), \ldots$ and count how many arrivals occur in each interval. Note that the number of arrivals in any small time interval of length h is, with high probability, either 1 (with probability $\lambda h \mathrm{e}^{-\lambda h} \approx \lambda h$) or 0 (with probability $\mathrm{e}^{-\lambda h} \approx 1 - \lambda h$). Next, define $X = \{X_n\}$ to be a Bernoulli process with success parameter $p = \lambda h$. Put $Y_0 = 0$ and let $Y_n = X_1 + \cdots + X_n$ be the total number of successes in n trials. $Y = \{Y_n\}$ is called the *Bernoulli approximation* to N. We can view N as a limiting case of Y as we decrease h.

As an example of the usefulness of this interpretation, we now demonstrate that the Poisson property (b) in Definition 1.11.1 follows basically from the *independence* assumption (a). For small h, N_t should have approximately the same distribution as Y_n, where n is the integer part of t/h (we write $n = \lfloor t/h \rfloor$). Hence,

$$\begin{aligned} \mathbb{P}(N_t = k) &\approx \mathbb{P}(Y_n = k) \\ &= \binom{n}{k} (\lambda h)^k (1 - (\lambda h))^{n-k} \\ &\approx \binom{n}{k} (\lambda t/n)^k (1 - (\lambda t/n))^{n-k} \\ &\approx \mathrm{e}^{\lambda t} \frac{(\lambda t)^k}{k!} . \end{aligned} \tag{1.29}$$

Equation (1.29) follows from the Poisson approximation to the binomial distribution; see Problem 1.22.

Another application of the Bernoulli approximation is the following. For the Bernoulli process, given that the total number of successes is k, the positions of the k successes are uniformly distributed over points $1, \ldots, n$. The corresponding property for the Poisson process N is that given $N_t = n$, the arrival times $T_1, \ldots, T_n$ are distributed according to the order statistics $X_{(1)}, \ldots, X_{(n)}$, where $X_1, \ldots, X_n$ are iid $\mathsf{U}[0, t]$.

1.12 MARKOV PROCESSES

Markov processes are stochastic processes whose futures are conditionally independent of their pasts given their present values. More formally, a stochastic process $\{X_t, t \in \mathscr{T}\}$, with $\mathscr{T} \subseteq \mathbb{R}$, is called a *Markov process* if, for every $s > 0$ and t,

$$(X_{t+s} \mid X_u, u \leqslant t) \sim (X_{t+s} \mid X_t) . \tag{1.30}$$

In other words, the conditional distribution of the future variable X_{t+s}, given the entire past of the process $\{X_u, u \leqslant t\}$, is the same as the conditional distribution of X_{t+s} given only the present X_t. That is, in order to predict future states, we only need to know the present one. Property (1.30) is called the *Markov property*.

Depending on the index set $\mathscr{T}$ and state space $\mathscr{E}$ (the set of all values the $\{X_t\}$ can take), Markov processes come in many different forms. A Markov process with a discrete index set is called a *Markov chain*. A Markov process with a discrete state space and a continuous index set (such as $\mathbb{R}$ or $\mathbb{R}_+$) is called a *Markov jump process*.

1.12.1 Markov Chains

Consider a Markov chain $X = \{X_t, t \in \mathbb{N}\}$ with a discrete (that is, countable) state space $\mathscr{E}$. In this case the Markov property (1.30) is:

$$\mathbb{P}(X_{t+1} = x_{t+1} \mid X_0 = x_0, \ldots, X_t = x_t) = \mathbb{P}(X_{t+1} = x_{t+1} \mid X_t = x_t) \tag{1.31}$$

for all $x_0, \ldots, x_{t+1}, \in \mathscr{E}$ and $t \in \mathbb{N}$. We restrict ourselves to Markov chains for which the conditional probability

$$\mathbb{P}(X_{t+1} = j \mid X_t = i),\ i, j \in \mathscr{E} \tag{1.32}$$

is independent of the time t. Such chains are called *time-homogeneous*. The probabilities in (1.32) are called the *(one-step) transition probabilities* of X. The distribution of X_0 is called the *initial distribution* of the Markov chain. The one-step transition probabilities and the initial distribution completely specify the distribution of X. Namely, we have by the product rule (1.4) and the Markov property (1.30)

$$\begin{aligned}
&\mathbb{P}(X_0 = x_0, \ldots, X_t = x_t) \\
&\quad = \mathbb{P}(X_0 = x_0)\, \mathbb{P}(X_1 = x_1 \mid X_0 = x_0) \cdots \mathbb{P}(X_t = x_t \mid X_0 = x_0, \ldots X_{t-1} = x_{t-1}) \\
&\quad = \mathbb{P}(X_0 = x_0)\, \mathbb{P}(X_1 = x_1 \mid X_0 = x_0) \cdots \mathbb{P}(X_t = x_t \mid X_{t-1} = x_{t-1})\,.
\end{aligned}$$

Since $\mathscr{E}$ is countable, we can arrange the one-step transition probabilities in an array. This array is called the (one-step) *transition matrix* of X. We usually denote it by P. For example, when $\mathscr{E} = \{0, 1, 2, \ldots\}$ the transition matrix P has the form

$$P = \begin{pmatrix} p_{00} & p_{01} & p_{02} & \cdots \\ p_{10} & p_{11} & p_{12} & \cdots \\ p_{20} & p_{21} & p_{22} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

Note that the elements in every row are positive and sum up to unity.

Another convenient way to describe a Markov chain X is through its *transition graph*. States are indicated by the nodes of the graph, and a strictly positive (> 0) transition probability p_{ij} from state i to j is indicated by an arrow from i to j with weight p_{ij}.

■ **EXAMPLE 1.10 Random Walk on the Integers**

Let p be a number between 0 and 1. The Markov chain X with state space $\mathbb{Z}$ and transition matrix P defined by

$$P(i, i+1) = p, \quad P(i, i-1) = q = 1 - p, \quad \text{for all } i \in \mathbb{Z}$$

is called a *random walk on the integers*. Let X start at 0; thus, $\mathbb{P}(X_0 = 0) = 1$. The corresponding transition graph is given in Figure 1.4. Starting at 0, the chain takes subsequent steps to the right with probability p and to the left with probability q.

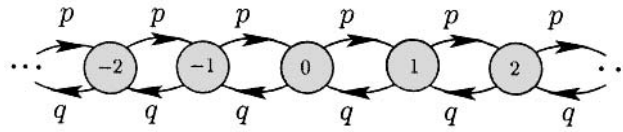


Figure 1.4 Transition graph for a random walk on $\mathbb{Z}$.

We shall show next how to calculate the probability that, starting from state i at some (discrete) time t, we are in j at (discrete) time $t+s$, that is, the probability $\mathbb{P}(X_{t+s} = j \mid X_t = i)$. For clarity, let us assume that $\mathscr{E} = \{1, 2, \ldots, m\}$ for some fixed m, so that P is an $m \times m$ matrix. For $t = 0, 1, 2, \ldots$, define the row vector

$$\boldsymbol{\pi}^{(t)} = (\mathbb{P}(X_t = 1), \ldots, \mathbb{P}(X_t = m)).$$

We call $\boldsymbol{\pi}^{(t)}$ the *distribution vector*, or simply the *distribution*, of X at time t and $\boldsymbol{\pi}^{(0)}$ the *initial distribution* of X. The following result shows that the t-step probabilities can be found simply by matrix multiplication.

Theorem 1.12.1 *The distribution of X at time t is given by*

$$\boldsymbol{\pi}^{(t)} = \boldsymbol{\pi}^{(0)} P^t \tag{1.33}$$

for all $t = 0, 1, \ldots$. (Here P^0 denotes the identity matrix.)

Proof: The proof is by induction. Equality (1.33) holds for $t = 0$ by definition. Suppose it is true for some $t = 0, 1, \ldots$. We have

$$\mathbb{P}(X_{t+1} = k) = \sum_{i=1}^{m} \mathbb{P}(X_{t+1} = k \mid X_t = i)\, \mathbb{P}(X_t = i)\ .$$

But (1.33) is assumed to be true for t, so $\mathbb{P}(X_t = i)$ is the i-th element of $\boldsymbol{\pi}^{(0)} P^t$. Moreover, $\mathbb{P}(X_{t+1} = k \mid X_t = i)$ is the (i,k)-th element of P. Therefore, for every k

$$\sum_{i=1}^{m} \mathbb{P}(X_{t+1} = k \mid X_t = i)\, \mathbb{P}(X_t = i) = \sum_{i=1}^{m} P(i,k)(\boldsymbol{\pi}^{(0)} P^t)(i)\ ,$$

which is just the k-th element of $\boldsymbol{\pi}^{(0)} P^{t+1}$. This completes the induction step, and thus the theorem is proved. □

By taking $\boldsymbol{\pi}^{(0)}$ as the i-th unit vector, $\mathbf{e}_i$, the t-step transition probabilities can be found as $\mathbb{P}(X_t = j \mid X_0 = i) = (\mathbf{e}_i P^t)(j) = P^t(i,j)$, which is the (i,j)-th element of matrix P^t. Thus, to find the t-step transition probabilities, we just have to compute the t-th power of P.

1.12.2 Classification of States

Let X be a Markov chain with discrete state space $\mathscr{E}$ and transition matrix P. We can characterize the relations between states in the following way: If states i and j are such that $P^t(i,j) > 0$ for some $t \geqslant 0$, we say that i *leads to* j and write $i \rightarrow j$. We say that i and j

communicate if $i \to j$ and $j \to i$, and write $i \leftrightarrow j$. Using the relation "$\leftrightarrow$" we can divide $\mathscr{E}$ into *equivalence classes* such that all the states in an equivalence class communicate with each other but not with any state outside that class. If there is only one equivalent class ($= \mathscr{E}$), the Markov chain is said to be *irreducible*. If a set of states $\mathscr{A}$ is such that $\sum_{j \in \mathscr{A}} P(i,j) = 1$ for all $i \in \mathscr{A}$, then $\mathscr{A}$ is called a *closed* set. A state i is called an *absorbing* state if $\{i\}$ is closed. For example, in the transition graph depicted in Figure 1.5, the equivalence classes are $\{1,2\}$, $\{3\}$, and $\{4,5\}$. Class $\{1,2\}$ is the only closed set: the Markov chain cannot escape from it. If state 1 were missing, state 2 would be absorbing. In Example 1.10 the Markov chain is irreducible since all states communicate.

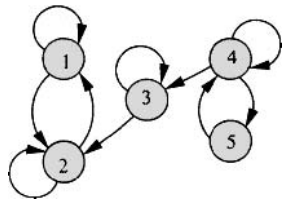


Figure 1.5 A transition graph with three equivalence classes.

Another classification of states is obtained by observing the system from a local point of view. In particular, let T denote the time the chain first visits state j, or first returns to j if it started there, and let N_j denote the total number of visits to j from time 0 on. We write $\mathbb{P}_j(A)$ for $\mathbb{P}(A \mid X_0 = j)$ for any event A. We denote the corresponding expectation operator by $\mathbb{E}_j$. State j is called a *recurrent* state if $\mathbb{P}_j(T < \infty) = 1$; otherwise, j is called *transient*. A recurrent state is called *positive recurrent* if $\mathbb{E}_j[T] < \infty$; otherwise, it is called *null recurrent*. Finally, a state is said to be *periodic, with period* δ, if $\delta \geqslant 2$ is the largest integer for which $\mathbb{P}_j(T = n\delta \text{ for some } n \geqslant 1) = 1$; otherwise, it is called *aperiodic*. For example, in Figure 1.5 states 1 and 2 are recurrent, and the other states are transient. All these states are aperiodic. The states of the random walk of Example 1.10 are periodic with period 2.

It can be shown that recurrence and transience are class properties. In particular, if $i \leftrightarrow j$, then i recurrent (transient) $\Leftrightarrow$ j recurrent (transient). Thus, in an irreducible Markov chain, one state being recurrent implies that all other states are also recurrent. And if one state is transient, then so are all the others.

1.12.3 Limiting Behavior

The limiting or "steady-state" behavior of Markov chains as $t \to \infty$ is of considerable interest and importance, and is often simpler to describe and analyze than the "transient" behavior of the chain for fixed t. It can be shown (see, for example, [4]) that in an irreducible, aperiodic Markov chain with transition matrix P the t-step probabilities converge to a constant that does not depend on the initial state. More specifically,

$$\lim_{t \to \infty} P^t(i,j) = \pi_j \tag{1.34}$$

for some number $0 \leqslant \pi_j \leqslant 1$. Moreover, $\pi_j > 0$ if j is positive recurrent and $\pi_j = 0$ otherwise. The intuitive reason behind this result is that the process "forgets" where it was initially if it goes on long enough. This is true for both finite and countably infinite Markov chains. The numbers $\{\pi_j, j \in \mathscr{E}\}$ form the *limiting distribution* of the Markov

chain, provided that $\pi_j \geqslant 0$ and $\sum_j \pi_j = 1$. Note that these conditions are not always satisfied: they are clearly not satisfied if the Markov chain is transient, and they may not be satisfied if the Markov chain is recurrent (namely when the states are null-recurrent). The following theorem gives a method for obtaining limiting distributions. Here we assume for simplicity that $\mathscr{E} = \{0, 1, 2, \ldots\}$. The limiting distribution is identified with the row vector $\pi = (\pi_0, \pi_1, \ldots)$.

Theorem 1.12.2 *For an irreducible, aperiodic Markov chain with transition matrix P, if the limiting distribution π exists, then it is uniquely determined by the solution of*

$$\pi = \pi P \,, \tag{1.35}$$

with $\pi_j \geqslant 0$ and $\sum_j \pi_j = 1$. Conversely, if there exists a positive row vector π satisfying (1.35) *and summing up to 1, then π is the limiting distribution of the Markov chain. Moreover, in that case $\pi_j > 0$ for all j, and all states are positive recurrent.*

Proof: (Sketch). For the case where $\mathscr{E}$ is finite, the result is simply a consequence of (1.33). Namely, with $\pi^{(0)}$ being the i-th unit vector, we have

$$P^{t+1}(i,j) = \left(\pi^{(0)}\, P^t\, P\right)(j) = \sum_{k \in \mathscr{E}} P^t(i,k) P(k,j) \,.$$

Letting $t \to \infty$, we obtain (1.35) from (1.34), provided that we can change the order of the limit and the summation. To show uniqueness, suppose that another vector $\mathbf{y}$, with $y_j \geqslant 0$ and $\sum_j y_j = 1$, satisfies $\mathbf{y} = \mathbf{y}P$. Then it is easy to show by induction that $\mathbf{y} = \mathbf{y}P^t$, for every t. Hence, letting $t \to \infty$, we obtain for every j

$$y_j = \sum_i y_i \pi_j = \pi_j \,,$$

since the $\{y_j\}$ sum up to unity. We omit the proof of the converse statement. □

■ **EXAMPLE 1.11 Random Walk on the Positive Integers**

This is a slightly different random walk than the one in Example 1.10. Let X be a random walk on $\mathscr{E} = \{0, 1, 2, \ldots\}$ with transition matrix

$$P = \begin{pmatrix} q & p & 0 & \ldots & & \\ q & 0 & p & 0 & \ldots & \\ 0 & q & 0 & p & 0 & \ldots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix},$$

where $0 < p < 1$ and $q = 1 - p$. X_t could represent, for example, the number of customers who are waiting in a queue at time t.

All states can be reached from each other, so the chain is irreducible and every state is either recurrent or transient. The equation $\pi = \pi P$ becomes

$$\begin{aligned} \pi_0 &= q\,\pi_0 + q\,\pi_1 \,, \\ \pi_1 &= p\,\pi_0 + q\,\pi_2 \,, \\ \pi_2 &= p\,\pi_1 + q\,\pi_3 \,, \\ \pi_3 &= p\,\pi_2 + q\,\pi_4 \,, \end{aligned}$$

and so on. We can solve this set of equation sequentially. If we let $r = p/q$, then we can express the $\pi_1, \pi_2, \ldots$ in terms of π_0 and r as

$$\pi_j = r^j \pi_0, \quad j = 0, 1, 2, \ldots .$$

If $p < q$, then $r < 1$ and $\sum_{j=0}^{\infty} \pi_j = \pi_0/(1-r)$, and by choosing $\pi_0 = 1 - r$, we can make the sum $\sum \pi_j = 1$. Hence, for $r < 1$ we have found the limiting distribution $\pi = (1-r)(1, r, r^2, r^3, \ldots)$ for this Markov chain, and all the states are therefore positive recurrent. On the other hand, when $p \geqslant q$, $\sum \pi_j$ is either 0 or infinite, and hence all states are either null-recurrent or transient. (It can be shown that only the case $p = q$ leads to null-recurrent states.)

Let X be a Markov chain with limiting distribution π. Suppose $\pi^{(0)} = \pi$. Then, combining (1.33) and (1.35), we have $\pi^{(t)} = \pi$. Thus, if the initial distribution of the Markov chain is equal to the limiting distribution, then the distribution of X_t is the same for all t (and is given by this limiting distribution). In fact, it is not difficult to show that for any k the distribution of $X_k, X_{k+1}, X_{k+2} \ldots$ is the same as that of $X_0, X_1, \ldots$. In other words, when $\pi^{(0)} = \pi$, the Markov chain is a stationary stochastic process. More formally, a stochastic process $\{X_t, t \in \mathbb{N}\}$ is called *stationary* if, for any positive $\tau, t_1, \ldots, t_n$, the vector $(X_{t_1}, \ldots, X_{t_n})$ has the same distribution as $(X_{t_1+\tau}, \ldots, X_{t_n+\tau})$. Similar definitions hold when the index set is $\mathbb{Z}$, $\mathbb{R}_+$ or $\mathbb{R}$. For this reason, any distribution π for which (1.35) holds is called a *stationary distribution*.

Noting that $\sum_j p_{ij} = 1$, we can rewrite (1.35) as the system of equations

$$\sum_j \pi_i p_{ij} = \sum_j \pi_j p_{ji} \quad \text{for all } i \in \mathscr{E} . \tag{1.36}$$

These are called the *global balance equations*. We can interpret (1.35) as the statement that the "probability flux" out of i is balanced by the probability flux into i. An important generalization, which follows directly from (1.36), states that the same balancing of probability fluxes holds for an arbitrary set $\mathscr{A}$. That is, for every set $\mathscr{A}$ of states we have

$$\sum_{i \in \mathscr{A}} \sum_{j \notin \mathscr{A}} \pi_i p_{ij} = \sum_{i \in \mathscr{A}} \sum_{j \notin \mathscr{A}} \pi_j p_{ji} . \tag{1.37}$$

1.12.4 Reversibility

Reversibility is an important notion in the theory of Markov and more general processes. A stationary stochastic process $\{X_t\}$ with index set $\mathbb{Z}$ or $\mathbb{R}$ is said to be *reversible* if, for any positive integer n and for all $t_1, \ldots, t_n$, the vector $(X_{t_1}, \ldots, X_{t_n})$ has the same distribution as $(X_{-t_1}, \ldots, X_{-t_n})$. One way to visualize this is to imagine that we have taken a video of the stochastic process, which we may run in forward and reverse time. If we cannot determine whether the video is running forward or backward, the process is reversible. The main result for reversible Markov chains is that a stationary Markov process is reversible if and only if there exists a collection of positive numbers $\{\pi_i, i \in \mathscr{E}\}$ summing to unity that satisfy the *detailed (or local) balance equations*

$$\pi_i p_{ij} = \pi_j p_{ji} , \quad i, j \in \mathscr{E}. \tag{1.38}$$

Whenever such a collection $\{\pi_j\}$ exists, it is the stationary distribution of the process.

A good way to think of the detailed balance equations is that they balance the probability flux from state i to state j with that from state j to state i. Contrast this with the equilibrium equations (1.36), which balance the probability flux out of state i with that into state i.

Kolmogorov's criterion is a simple criterion for reversibility based on the transition probabilities. It states that a stationary Markov process is reversible if and only if its transition rates satisfy

$$p(i_1, i_2)\, p(i_2, i_3) \cdots p(i_{n-1}, i_n)\, p(i_n, i_1) = p(i_1, i_n)\, p(i_n, i_{n-1}) \cdots p(i_2, i_1) \qquad (1.39)$$

for all finite loops of states $i_1, \ldots, i_n, i_1$. (For clarity, we have used the notation $p(i, j)$ rather than p_{ij} for the transition probabilities.) The idea is quite intuitive: if the process in forward time is more likely to traverse a certain closed loop in one direction than in the opposite direction, then in backward time it will exhibit the opposite behavior, and hence we have a criterion for detecting the direction of time. If such "looping" behavior does not occur, the process must be reversible.

1.12.5 Markov Jump Processes

A *Markov jump process* $X = \{X_t, t \geqslant 0\}$ can be viewed as a continuous-time generalization of a Markov chain and also of a Poisson process. The Markov property (1.30) now reads

$$\mathbb{P}(X_{t+s} = x_{t+s} \mid X_u = x_u, u \leqslant t) = \mathbb{P}(X_{t+s} = x_{t+s} \mid X_t = x_t)\,. \qquad (1.40)$$

As in the Markov chain case, one usually assumes that the process is *time-homogeneous*, that is, $\mathbb{P}(X_{t+s} = j \mid X_t = i)$ does not depend on t. Denote this probability by $P_s(i, j)$. An important quantity is the *transition rate* q_{ij} from state i to j, defined for $i \neq j$ as

$$q_{ij} = \lim_{t \downarrow 0} \frac{P_t(i, j)}{t}\,.$$

The sum of the rates out of state i is denoted by q_i. A typical sample path of X is shown in Figure 1.6. The process jumps at times $T_1, T_2, \ldots$ to states $Y_1, Y_2, \ldots$, staying some length of time in each state.

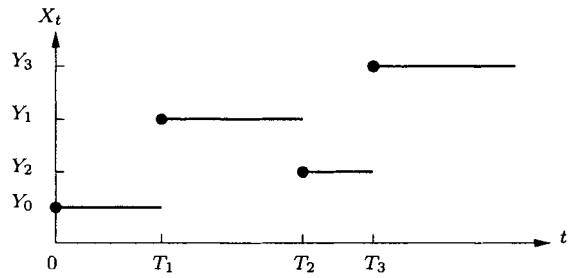


Figure 1.6 A sample path of a Markov jump process $\{X_t, t \geqslant 0\}$.

More precisely, a Markov jump process X behaves (under suitable regularity conditions; see [4]) as follows:

1. Given its past, the probability that X jumps from its current state i to state j is $K_{ij} = q_{ij}/q_i$.
2. The amount of time that X spends in state j has an exponential distribution with mean $1/q_j$, independent of its past history.

The first statement implies that the process $\{Y_n\}$ is in fact a Markov chain, with transition matrix $K = (K_{ij})$.

A convenient way to describe a Markov jump process is through its *transition rate graph*. This is similar to a transition graph for Markov chains. The states are represented by the nodes of the graph, and a transition rate from state i to j is indicated by an arrow from i to j with weight q_{ij}.

■ **EXAMPLE 1.12 Birth and Death Process**

A *birth and death process* is a Markov jump process with a transition rate graph of the form given in Figure 1.7. Imagine that X_t represents the total number of individuals in a population at time t. Jumps to the right correspond to births, and jumps to the left to deaths. The *birth rates* $\{b_i\}$ and the *death rates* $\{d_i\}$ may differ from state to state. Many applications of Markov chains involve processes of this kind. Note

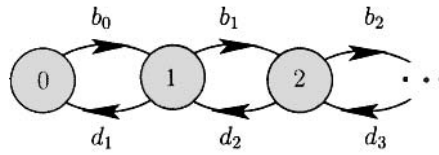


Figure 1.7 The transition rate graph of a birth and death process.

that the process jumps from one state to the next according to a Markov chain with transition probabilities $K_{0,1} = 1$, $K_{i,i+1} = b_i/(b_i + d_i)$, and $K_{i,i-1} = d_i/(b_i + d_i)$, $i = 1, 2, \ldots$. Moreover, it spends an $\mathsf{Exp}(b_0)$ amount of time in state 0 and $\mathsf{Exp}(b_i + d_i)$ in the other states.

Limiting Behavior We now formulate the continuous-time analogues of (1.34) and Theorem 1.12.2. Irreducibility and recurrence for Markov jump processes are defined in the same way as for Markov chains. For simplicity, we assume that $\mathscr{E} = \{1, 2, \ldots\}$. If X is a recurrent and irreducible Markov jump process, then irrespective of i,

$$\lim_{t \to \infty} \mathbb{P}(X_t = j \mid X_0 = i) = \pi_j \tag{1.41}$$

for some number $\pi_j \geqslant 0$. Moreover, $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots)$ is the solution to

$$\sum_{j \neq i} \pi_i \, q_{ij} = \sum_{j \neq i} \pi_j \, q_{ji}, \quad \text{for all } i = 1, \ldots, m \tag{1.42}$$

with $\sum_j \pi_j = 1$, if such a solution exists, in which case all states are positive recurrent. If such a solution does not exist, all π_j are 0.

As in the Markov chain case, $\{\pi_j\}$ is called the *limiting distribution* of X and is usually identified with the row vector π. Any solution π of (1.42) with $\sum_j \pi_j = 1$ is called a *stationary distribution*, since taking it as the initial distribution of the Markov jump process renders the process stationary.

The equations (1.42) are again called the *global balance equations* and are readily generalized to (1.37), replacing the transition probabilities with transition rates. More importantly, if the process is reversible, then, as with Markov chains, the stationary distribution can be found from the *local balance equations*:

$$\pi_i q_{ij} = \pi_j q_{ji} , \quad i, j \in \mathscr{E} . \tag{1.43}$$

Reversibility can be easily verified by checking that looping does not occur, that is, via Kolmogorov's criterion (1.39), replacing the probabilities p with rates q.

■ **EXAMPLE 1.13 $M/M/1$ Queue**

Consider a service facility where customers arrive at certain random times and are served by a single server. Arriving customers who find the server busy wait in the queue. Customers are served in the order in which they arrive. The interarrival times are exponential random variables with rates λ, and the service times of customers are iid exponential random variables with rates μ. Finally, the service times are independent of the interarrival times. Let X_t be the number of customers in the system at time t. By the memoryless property of the exponential distribution (see Problem 1.7), it is not difficult to see that $X = \{X_t, t \geqslant 0\}$ is a Markov jump process, and in fact a birth and death process with birth rates $b_i = \lambda$, $i = 0, 1, 2, \ldots$ and death rates $d_i = \mu$, $i = 1, 2, \ldots$.

Solving the global balance equations (or, more easily, the local balance equations, since X is reversible), we see that X has a limiting distribution given by

$$\lim_{t\to\infty} \mathbb{P}(X_t = n) = (1 - \varrho)\,\varrho^n, \quad n = 0, 1, 2, \ldots, \tag{1.44}$$

provided that $\varrho = \lambda/\mu < 1$. This means that the expected service time needs to be less than the expected interarrival time for a limiting distribution to exist. In that case, the limiting distribution is also the stationary distribution. In particular, if X_0 is distributed according to (1.44), then X_t has the same distribution for all $t > 0$.

1.13 EFFICIENCY OF ESTIMATORS

In this book we will frequently use

$$\widehat{\ell} = \frac{1}{N} \sum_{i=1}^{N} Z_i , \tag{1.45}$$

which presents an *unbiased* estimator of the unknown quantity $\ell = \mathbb{E}[\widehat{\ell}] = \mathbb{E}[Z]$, where $Z_1, \ldots, Z_N$ are independent replications of some random variable Z.

By the central limit theorem, $\widehat{\ell}$ has approximately a $\mathsf{N}(\ell, N^{-1}\mathrm{Var}(Z))$ distribution for large N. We shall estimate $\mathrm{Var}(Z)$ via the *sample variance*

$$S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (Z_i - \widehat{\ell})^2 .$$

By the law of large numbers, S^2 converges with probability 1 to $\mathrm{Var}(Z)$. Consequently, for $\mathrm{Var}(Z) < \infty$ and large N, the approximate $(1-\alpha)$ confidence interval for ℓ is given by

$$\left(\widehat{\ell} - z_{1-\alpha/2} \frac{S}{\sqrt{N}},\ \widehat{\ell} + z_{1-\alpha/2} \frac{S}{\sqrt{N}} \right) ,$$

where $z_{1-\alpha/2}$ is the $(1-\alpha/2)$ quantile of the standard normal distribution. For example, for $\alpha = 0.05$ we have $z_{1-\alpha/2} = z_{0.975} = 1.96$. The quantity

$$\frac{S/\sqrt{N}}{\widehat{\ell}}$$

is often used in the simulation literature as an accuracy measure for the estimator $\widehat{\ell}$. For large N it converges to the *relative error* of $\widehat{\ell}$, defined as

$$\kappa = \frac{\sqrt{\mathrm{Var}(\widehat{\ell})}}{\mathbb{E}[\widehat{\ell}]} = \frac{\sqrt{\mathrm{Var}(Z)/N}}{\ell} . \tag{1.46}$$

The square of the relative error

$$\kappa^2 = \frac{\mathrm{Var}(\widehat{\ell})}{\ell^2} \tag{1.47}$$

is called the *squared coefficient of variation*.

■ **EXAMPLE 1.14 Estimation of Rare-Event Probabilities**

Consider estimation of the tail probability $\ell = \mathbb{P}(X \geqslant \gamma)$ of some random variable X for a *large* number γ. If ℓ is very small, then the event $\{X \geqslant \gamma\}$ is called a *rare event* and the probability $\mathbb{P}(X \geqslant \gamma)$ is called a *rare-event probability*.

We may attempt to estimate ℓ via (1.45) as

$$\widehat{\ell} = \frac{1}{N} \sum_{i=1}^{N} I_{\{X_i \geqslant \gamma\}} , \tag{1.48}$$

which involves drawing a random sample $X_1, \ldots, X_N$ from the pdf of X and defining the indicators $Z_i = I_{\{X_i \geqslant \gamma\}}$, $i = 1, \ldots, N$. The estimator ℓ thus defined is called the *crude Monte Carlo* (CMC) estimator. For small ℓ the relative error of the CMC estimator is given by

$$\kappa = \frac{\sqrt{\mathrm{Var}(\widehat{\ell})}}{\mathbb{E}[\widehat{\ell}]} = \sqrt{\frac{1-\ell}{N\ell}} \approx \sqrt{\frac{1}{N\ell}} . \tag{1.49}$$

As a numerical example, suppose that $\ell = 10^{-6}$. In order to estimate ℓ accurately with relative error (say) $\kappa = 0.01$, we need to choose a sample size

$$N \approx \frac{1}{\kappa^2 \ell} = 10^{10} .$$

This shows that estimating small probabilities via CMC estimators is computationally meaningless.

1.13.1 Complexity

The theoretical framework in which one typically examines rare-event probability estimation is based on *complexity theory*, as introduced in [1, 12]. In particular, the estimators are classified either as *polynomial-time* or as *exponential-time*. It is shown in [1, 15] that for an arbitrary estimator, $\widehat{\ell}$ of ℓ, to be polynomial-time as a function of some γ, it suffices that its squared coefficient of variation, κ^2, or its relative error, κ, is bounded in γ by some polynomial function, $p(\gamma)$. For such polynomial-time estimators, the required sample size to achieve a fixed relative error does not grow too fast as the event becomes rarer.

Consider the estimator (1.48) and assume that ℓ becomes very small as $\gamma \to \infty$. Note that

$$\mathbb{E}[Z^2] \geqslant (\mathbb{E}[Z])^2 = \ell^2 .$$

Hence, the best one can hope for with such an estimator is that its second moment of Z^2 decreases proportionally to ℓ^2 as $\gamma \to \infty$. We say that the rare-event estimator (1.48) has *bounded relative error* if for all γ

$$\mathbb{E}[Z^2] \leqslant c\,\ell^2 \tag{1.50}$$

for some fixed $c \geqslant 1$. Because bounded relative error is not always easy to achieve, the following weaker criterion is often used. We say that the estimator (1.48) is *logarithmically efficient* (sometimes called *asymptotically optimal*) if

$$\lim_{\gamma\to\infty} \frac{\ln \mathbb{E}[Z^2]}{\ln \ell^2} = 1 . \tag{1.51}$$

■ **EXAMPLE 1.15 The CMC Estimator Is Not Logarithmically Efficient**

Consider the CMC estimator (1.48). We have

$$\mathbb{E}[Z^2] = \mathbb{E}[Z] = \ell ,$$

so that

$$\lim_{\gamma\to\infty} \frac{\ln \mathbb{E}[Z^2]}{\ln \ell^2(\gamma)} = \frac{\ln \ell}{\ln \ell^2} = \frac{1}{2} .$$

Hence, the CMC estimator is not logarithmically efficient, and therefore alternative estimators must be found to estimate small ℓ.

1.14 INFORMATION

In this section we discuss briefly various measures of information in a random experiment. Suppose we describe the measurements in a random experiment via a random vector $\mathbf{X} = (X_1, \ldots, X_n)$ with pdf f. Then all the information about the experiment (all of our probabilistic knowledge) is obviously contained in the pdf f. However, in most cases we wish to characterize our information about the experiments with just a few key numbers, such as the *expectation* and the *covariance matrix* of $\mathbf{X}$, which provide information about the mean measurements and the variability of the measurements, respectively. Another informational measure comes from coding and communications theory, where the *Shannon entropy* characterizes the average number of bits needed to transmit a message $\mathbf{X}$ over a (binary) communication channel. Yet another approach to information can be found in

statistics. Specifically, in the theory of point estimation, the pdf f depends on a parameter vector $\boldsymbol{\theta}$. The question is how well $\boldsymbol{\theta}$ can be estimated via an outcome of $\mathbf{X}$ — in other words, how much information about $\boldsymbol{\theta}$ is contained in the "data" $\mathbf{X}$. Various measures for this type of information are associated with the *maximum likelihood*, the *score*, and the *(Fisher) information matrix*. Finally, the amount of information in a random experiment can often be quantified via a *distance* concept, such as the *Kullback–Leibler* "distance" (divergence), also called the *cross-entropy*.

1.14.1 Shannon Entropy

One of the most celebrated measures of uncertainty in information theory is the *Shannon entropy*, or simply *entropy*. A good reference is [5], where the entropy of a discrete random variable X with density f is defined as

$$\mathbb{E}\left[\log_2 \frac{1}{f(X)}\right] = -\mathbb{E}\left[\log_2 f(X)\right] = -\sum_{\mathscr{X}} f(x) \log_2 f(x) \,.$$

Here X is interpreted as a random character from an alphabet $\mathscr{X}$, such that $X = x$ with probability $f(x)$. We will use the convention $0 \ln 0 = 0$.

It can be shown that the most efficient way to transmit characters sampled from f over a binary channel is to encode them such that the number of bits required to transmit x is equal to $\log_2(1/f(x))$. It follows that $-\sum_{\mathscr{X}} f(x) \log_2 f(x)$ is the expected bit length required to send a random character $X \sim f$; see [5].

A more general approach, which includes continuous random variables, is to define the entropy of a random variable X with density f by

$$\mathcal{H}(X) = -\mathbb{E}[\ln f(X)] = \begin{cases} -\sum f(x) \ln f(x) & \text{discrete case,} \\ -\int f(x) \ln f(x)\, dx & \text{continuous case.} \end{cases} \tag{1.52}$$

Definition (1.52) can easily be extended to random vectors $\mathbf{X}$ as (in the continuous case)

$$\mathcal{H}(\mathbf{X}) = -\mathbb{E}[\ln f(\mathbf{X})] = -\int f(\mathbf{x}) \ln f(\mathbf{x})\, d\mathbf{x} \,. \tag{1.53}$$

Often $\mathcal{H}(\mathbf{X})$ is called the *joint* entropy of the random variables $X_1, \ldots, X_n$ and is also written as $\mathcal{H}(X_1, \ldots, X_n)$. In the continuous case, $\mathcal{H}(\mathbf{X})$ is frequently referred to as the *differential entropy* to distinguish it from the discrete case.

■ **EXAMPLE 1.16**

Let X have a Ber(p) distribution for some $0 \leqslant p \leqslant 1$. The density f of X is given by $f(1) = \mathbb{P}(X = 1) = p$ and $f(0) = \mathbb{P}(X = 0) = 1 - p$, so that the entropy of X is

$$\mathcal{H}(X) = -p \ln p - (1 - p) \ln(1 - p) \,.$$

The graph of the entropy as a function of p is depicted in Figure 1.8. Note that the entropy is maximal for $p = 1/2$, which gives the "uniform" density on $\{0, 1\}$.

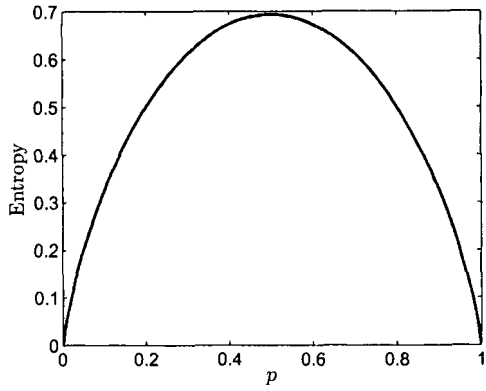


Figure 1.8 The entropy for the Ber(p) distribution as a function of p.

Next, consider a sequence $X_1, \ldots, X_n$ of iid Ber(p) random variables. Let $\mathbf{X} = (X_1, \ldots, X_n)$. The density of $\mathbf{X}$, say g, is simply the product of the densities of the X_i, so that

$$\mathcal{H}(\mathbf{X}) = -\mathbb{E}\left[\ln g(\mathbf{X})\right] = -\mathbb{E}\left[\ln \prod_{i=1}^{n} f(X_i)\right] = \sum_{i=1}^{n} -\mathbb{E}\left[\ln f(X_i)\right] = n\,\mathcal{H}(X) .$$

The properties of $\mathcal{H}(\mathbf{X})$ in the continuous case are somewhat different from those in the discrete one. In particular:

1. The differential entropy can be negative, whereas the discrete entropy is always positive.
2. The discrete entropy is insensitive to invertible transformations, whereas the differential entropy is not. Specifically, if $\mathbf{X}$ is discrete, $\mathbf{Y} = g(\mathbf{X})$, and g is an invertible mapping, then $\mathcal{H}(\mathbf{X}) = \mathcal{H}(\mathbf{Y})$, because $f_{\mathbf{Y}}(\mathbf{y}) = f_{\mathbf{X}}(g^{-1}(\mathbf{y}))$. However, in the continuous case, we have an additional factor due to the Jacobian of the transformation.

It is not difficult to see that of any density f, the one that gives the maximum entropy is the uniform density on $\mathscr{X}$. That is,

$$\mathcal{H}(\mathbf{X}) \text{ is maximal } \Leftrightarrow \; f(\mathbf{x}) = \frac{1}{|\mathscr{X}|} \text{ (constant)} . \tag{1.54}$$

For two random vectors $\mathbf{X}$ and $\mathbf{Y}$ with joint pdf f, we define the *conditional entropy* of $\mathbf{Y}$ given $\mathbf{X}$ as

$$\mathcal{H}(\mathbf{Y} \mid \mathbf{X}) = -\mathbb{E}\left[\ln \frac{f(\mathbf{X}, \mathbf{Y})}{f_{\mathbf{X}}(\mathbf{X})}\right] = \mathcal{H}(\mathbf{X}, \mathbf{Y}) - \mathcal{H}(\mathbf{X}) , \tag{1.55}$$

where $f_{\mathbf{X}}$ is the pdf of $\mathbf{X}$ and $\frac{f(\mathbf{x},\mathbf{y})}{f_{\mathbf{X}}(\mathbf{x})}$ is the conditional density of $\mathbf{Y}$ (at $\mathbf{y}$), given $\mathbf{X} = \mathbf{x}$. It follows that

$$\mathcal{H}(\mathbf{X}, \mathbf{Y}) = \mathcal{H}(\mathbf{X}) + \mathcal{H}(\mathbf{Y} \mid \mathbf{X}) = \mathcal{H}(\mathbf{Y}) + \mathcal{H}(\mathbf{X} \mid \mathbf{Y}) . \tag{1.56}$$

It is reasonable to impose that any sensible additive measure describing the average amount of uncertainty should satisfy at least (1.56) and (1.54). It follows that the uniform density carries the least amount of information, and the entropy (average amount of uncertainty) of $(\mathbf{X}, \mathbf{Y})$ is equal to the sum of the entropy of $\mathbf{X}$ and the amount of entropy in $\mathbf{Y}$ after the information in $\mathbf{X}$ has been accounted for. It is argued in [11] that any concept of entropy that includes the general properties (1.54) and (1.56) must lead to the definition (1.53).

The *mutual information* of $\mathbf{X}$ and $\mathbf{Y}$ is defined as

$$\mathcal{M}(\mathbf{X}, \mathbf{Y}) = \mathcal{H}(\mathbf{X}) + \mathcal{H}(\mathbf{Y}) - \mathcal{H}(\mathbf{X}, \mathbf{Y}) , \tag{1.57}$$

which, as the name suggests, can be interpreted as the amount of information shared by $\mathbf{X}$ and $\mathbf{Y}$. An alternative expression, which follows from (1.56) and (1.57), is

$$\mathcal{M}(\mathbf{X}, \mathbf{Y}) = \mathcal{H}(\mathbf{X}) - \mathcal{H}(\mathbf{X} \mid \mathbf{Y}) = \mathcal{H}(\mathbf{Y}) - \mathcal{H}(\mathbf{Y} \mid \mathbf{X}) , \tag{1.58}$$

which can be interpreted as the reduction of the uncertainty of one random variable due to the knowledge of the other. It is not difficult to show that the mutual information is always positive. It is also related to the cross-entropy concept, which follows.

1.14.2 Kullback–Leibler Cross-Entropy

Let g and h be two densities on $\mathscr{X}$. The Kullback–Leibler cross-entropy between g and h (compare with (1.53)) is defined (in the continuous case) as

$$\begin{aligned} \mathcal{D}(g, h) &= \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] \\ &= \int g(\mathbf{x}) \ln g(\mathbf{x}) \, d\mathbf{x} - \int g(\mathbf{x}) \ln h(\mathbf{x}) \, d\mathbf{x} . \end{aligned} \tag{1.59}$$

$\mathcal{D}(g, h)$ is also called the *Kullback–Leibler divergence*, the *cross-entropy*, and the *relative entropy*. If not stated otherwise, we shall call $\mathcal{D}(g, h)$ the *cross-entropy* (CE) between g and h. Notice that $\mathcal{D}(g, h)$ is not a distance between g and h in the formal sense, since in general $\mathcal{D}(g, h) \neq \mathcal{D}(h, g)$. Nonetheless, it is often useful to think of $\mathcal{D}(g, h)$ as a distance because

$$\mathcal{D}(g, h) \geqslant 0$$

and $\mathcal{D}(g, h) = 0$ if and only if $g(x) = h(x)$. This follows from Jensen's inequality (if ϕ is a convex function, such as $-\ln$, then $\mathbb{E}[\phi(X)] \geqslant \phi(\mathbb{E}[X])$). Namely,

$$\mathcal{D}(g, h) = \mathbb{E}_g \left[-\ln \frac{h(\mathbf{X})}{g(\mathbf{X})} \right] \geqslant -\ln \left\{ \mathbb{E}_g \left[\frac{h(\mathbf{X})}{g(\mathbf{X})} \right] \right\} = -\ln 1 = 0 .$$

It can be readily seen that the mutual information $\mathcal{M}(\mathbf{X}, \mathbf{Y})$ of vectors $\mathbf{X}$ and $\mathbf{Y}$ defined in (1.57) is related to the CE in the following way:

$$\mathcal{M}(\mathbf{X}, \mathbf{Y}) = \mathcal{D}(f, f_{\mathbf{X}} f_{\mathbf{Y}}) = \mathbb{E}_f \left[\ln \frac{f(\mathbf{X}, \mathbf{Y})}{f_{\mathbf{X}}(\mathbf{X}) \, f_{\mathbf{Y}}(\mathbf{Y})} \right] ,$$

where f is the (joint) pdf of $(\mathbf{X}, \mathbf{Y})$ and $f_{\mathbf{X}}$ and $f_{\mathbf{Y}}$ are the (marginal) pdfs of $\mathbf{X}$ and $\mathbf{Y}$, respectively. In other words, the mutual information can be viewed as the CE that measures

the distance between the joint pdf f of $\mathbf{X}$ and $\mathbf{Y}$ and the product of their marginal pdfs $f_{\mathbf{X}}$ and $f_{\mathbf{Y}}$, that is, under the assumption that the vectors $\mathbf{X}$ and $\mathbf{Y}$ are *independent*.

Remark 1.14.1 (Other Distance Measures) Instead of Kullback–Leibler distance, one can use several other distance or divergence measures between pdfs. An important class of such "distances" is formed by Csiszár's *ϕ-divergence* [10],

$$d(g,\, h) = \int_{\mathscr{X}} p(\mathbf{x})\, \phi\left(\frac{g(\mathbf{x})}{h(\mathbf{x})}\right) d\mathbf{x}\,, \tag{1.60}$$

where ϕ is any function such that $\phi(1) = 0$ and $\phi''(x) > 0$, $x > 0$ (in particular, ϕ is convex). Below is a list of important divergence measures that can be found as special cases of the ϕ-divergence.

- *Burg CE distance*:

$$d(g,\, h) = \int h(\mathbf{x}) \ln \frac{h(\mathbf{x})}{g(\mathbf{x})}\, d\mathbf{x}$$

- *Kullback–Leibler CE distance*:

$$d(g,\, h) = \int g(\mathbf{x}) \ln \frac{g(\mathbf{x})}{h(\mathbf{x})}\, d\mathbf{x}$$

- *Hellinger distance*:

$$d(g,\, h) = 2 \int \left(\sqrt{g(\mathbf{x})} - \sqrt{h(\mathbf{x})}\right)^2 d\mathbf{x}$$

- *Pearson χ^2 discrepancy measure*:

$$d(g,\, h) = \frac{1}{2} \int \frac{\left[g(\mathbf{x}) - h(\mathbf{x})\right]^2}{h(\mathbf{x})}\, d\mathbf{x}$$

- *Neymann χ^2 goodness of fit measure*:

$$d(g,\, h) = \frac{1}{2} \int \frac{\left[g(\mathbf{x}) - h(\mathbf{x})\right]^2}{g(\mathbf{x})}\, d\mathbf{x}$$

1.14.3 The Maximum Likelihood Estimator and the Score Function

We introduce here the notion of the *score function* (SF) via the classical *maximum likelihood estimator*. Consider a random vector $\mathbf{X} = (X_1, \ldots, X_n)$, which is distributed according to a fixed pdf $f(\cdot; \boldsymbol{\theta})$ with unknown parameter (vector) $\boldsymbol{\theta} \in \Theta$. Assume that we wish to estimate $\boldsymbol{\theta}$ on the basis of a given outcome $\mathbf{x}$ (the data) of $\mathbf{X}$. For a given $\mathbf{x}$, the function $\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}) = f(\mathbf{x}; \boldsymbol{\theta})$ is called the *likelihood function*. Note that $\mathcal{L}$ is a function of $\boldsymbol{\theta}$ for a fixed parameter $\mathbf{x}$, whereas for the pdf f it is the other way around. The maximum likelihood *estimate* $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\theta}}(\mathbf{x})$ of $\boldsymbol{\theta}$ is defined as

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}}\, \mathcal{L}(\boldsymbol{\theta}; \mathbf{x})\,. \tag{1.61}$$

Because the function ln is monotone increasing, we also have

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}} \ln \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}) \,. \tag{1.62}$$

The random variable $\widehat{\boldsymbol{\theta}}(\mathbf{X})$ with $\mathbf{X} \sim f(\cdot; \boldsymbol{\theta})$ is the corresponding maximum likelihood *estimator*, which is again written as $\widehat{\boldsymbol{\theta}}$. Note that often the data $X_1, \ldots, X_n$ form a random sample from some pdf $f_1(\cdot; \boldsymbol{\theta})$, in which case $f(\mathbf{x}; \boldsymbol{\theta}) = \prod_{i=1}^{N} f_1(x_i; \boldsymbol{\theta})$ and

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}} \sum_{i=1}^{N} \ln f_1(X_i; \boldsymbol{\theta}) \,. \tag{1.63}$$

If $\mathcal{L}(\boldsymbol{\theta}; \mathbf{x})$ is a continuously differentiable concave function with respect to $\boldsymbol{\theta}$ and the maximum is attained in the interior of Θ, then we can find the maximum likelihood estimator of $\boldsymbol{\theta}$ by solving

$$\nabla_{\boldsymbol{\theta}} \ln \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}) = \mathbf{0} \,.$$

The function $\mathcal{S}(\cdot; \mathbf{x})$ defined by

$$\mathcal{S}(\boldsymbol{\theta}; \mathbf{x}) = \nabla_{\boldsymbol{\theta}} \ln \mathcal{L}(\boldsymbol{\theta}; \mathbf{x}) = \frac{\nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta})}{f(\mathbf{x}; \boldsymbol{\theta})} \tag{1.64}$$

is called the *score function*. For the exponential family (A.9) it is easy to see that

$$\mathcal{S}(\boldsymbol{\theta}; \mathbf{x}) = \frac{\nabla c(\boldsymbol{\theta})}{c(\boldsymbol{\theta})} + \mathbf{t}(\mathbf{x}) \,. \tag{1.65}$$

The *random vector* $\mathcal{S}(\boldsymbol{\theta}) = \mathcal{S}(\boldsymbol{\theta}; \mathbf{X})$ with $\mathbf{X} \sim f(\cdot; \boldsymbol{\theta})$ is called the *(efficient) score*. The expected score is always equal to the zero vector, that is

$$\mathbb{E}_{\boldsymbol{\theta}}[\mathcal{S}(\boldsymbol{\theta})] = \int \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}) \, \mu(d\mathbf{x}) = \nabla_{\boldsymbol{\theta}} \int f(\mathbf{x}; \boldsymbol{\theta}) \, \mu(d\mathbf{x}) = \nabla_{\boldsymbol{\theta}} 1 = \mathbf{0} \,,$$

where the interchange of differentiation and integration is justified via the bounded convergence theorem.

1.14.4 Fisher Information

The covariance matrix $\mathcal{I}(\boldsymbol{\theta})$ of the score $\mathcal{S}(\boldsymbol{\theta})$ is called the *Fisher information matrix*. Since the expected score is always $\mathbf{0}$, we have

$$\mathcal{I}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}} \left[\mathcal{S}(\boldsymbol{\theta}) \mathcal{S}(\boldsymbol{\theta})^T \right] \,. \tag{1.66}$$

In the one-dimensional case we thus have

$$\mathcal{I}(\theta) = \mathbb{E}_{\theta} \left[\left(\frac{\partial \ln f(X; \theta)}{\partial \theta} \right)^2 \right] \,.$$

Because

$$\frac{\partial^2}{\partial \theta^2} \ln f(x; \theta) = \frac{\dfrac{\partial^2}{\partial \theta^2} f(x; \theta)}{f(x; \theta)} - \left(\frac{\dfrac{\partial}{\partial \theta} f(x; \theta)}{f(x; \theta)} \right)^2 \,,$$

we see that (under straightforward regularity conditions) the Fisher information is also given by

$$\mathcal{I}(\theta) = -\mathbb{E}_\theta \left[\frac{\partial^2 \ln f(X;\theta)}{\partial \theta^2} \right] .$$

In the multidimensional case we have similarly

$$\mathcal{I}(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{\theta}} \left[\nabla \mathcal{S}(\boldsymbol{\theta}) \right] = -\mathbb{E}_{\boldsymbol{\theta}} \left[\nabla^2 \ln f(\mathbf{X};\boldsymbol{\theta}) \right] , \tag{1.67}$$

where $\nabla^2 \ln f(\mathbf{X};\boldsymbol{\theta})$ denotes the *Hessian* of $\ln f(\mathbf{X};\boldsymbol{\theta})$, that is, the (random) matrix

$$\left(\frac{\partial^2 \ln f(\mathbf{X};\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j} \right) .$$

The importance of the Fisher information in statistics is corroborated by the famous *Cramér-Rao inequality*, which (in a simplified form) states that the variance of any unbiased estimator Z of $g(\boldsymbol{\theta})$ is bounded from below via

$$\mathrm{Var}(Z) \geqslant (\nabla g(\boldsymbol{\theta}))^T \, \mathcal{I}^{-1}(\boldsymbol{\theta}) \, \nabla g(\boldsymbol{\theta}) . \tag{1.68}$$

For more details see [13].

1.15 CONVEX OPTIMIZATION AND DUALITY

Let $f(x)$, $x \in \mathbb{R}$, be a real-valued function with continuous derivatives — also called a C^1 function. The standard approach to minimizing $f(x)$ is to solve the equation

$$f'(x) = 0 . \tag{1.69}$$

The solutions to (1.69) are called *stationary points*. If, in addition, the function has continuous second derivatives (a so-called C^2 function), the condition

$$f''(x^*) > 0 \tag{1.70}$$

ensures that a stationary point x^* is a *local minimizer*, that is, $f(x^*) < f(x)$ for all x in a small enough neighborhood of x^*.

For a C^1 function on $\mathbb{R}^n$, (1.69) generalizes to

$$\nabla f(\mathbf{x}) \equiv \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix} = \mathbf{0} , \tag{1.71}$$

where $\nabla f(\mathbf{x})$ is the *gradient* of f at $\mathbf{x}$. Similarly, a stationary point $\mathbf{x}^*$ is a local minimizer of f if the *Hessian matrix* (or simply *Hessian*) at $\mathbf{x}^*$,

$$\nabla^2 f(\mathbf{x}^*) \equiv \begin{pmatrix} \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_1 \partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f(\mathbf{x}^*)}{\partial x_n^2} \end{pmatrix} , \tag{1.72}$$

is *positive definite*, that is, $\mathbf{x}^T \left[\nabla^2 f(\mathbf{x}^*) \right] \mathbf{x} > 0$ for all $\mathbf{x} \neq \mathbf{0}$.

The situation can be further generalized by introducing *constraints*. A general constrained optimization problems can be written as

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x}) \tag{1.73}$$

$$\text{subject to:} \quad h_i(\mathbf{x}) = 0, \quad i = 1, \ldots, m \tag{1.74}$$

$$g_i(\mathbf{x}) \leqslant 0, \quad i = 1, \ldots, k. \tag{1.75}$$

Here f, g_i and h_i are given functions, $f(\mathbf{x})$ is called the *objective function*, and $h_i(\mathbf{x}) = 0$ and $g_i(\mathbf{x}) \leqslant 0$ represent the *equality* and *inequality* constraints, respectively.

The region of the domain where the objective function is defined and where all the constraints are satisfied is called the *feasible region*. A *global solution* to the optimization problem is a point $\mathbf{x}^* \in \mathbb{R}^n$ such that there exists no other point $\mathbf{x} \in \mathbb{R}^n$ for which $f(\mathbf{x}) < f(\mathbf{x}^*)$. Alternative names are *global minimizer* and *global minimum*, although the latter could be confused with the minimum value of the function. Similarly, for a *local* solution/minimizer, the condition $f(\mathbf{x}) < f(\mathbf{x}^*)$ only needs to hold in some neighborhood of $\mathbf{x}^*$.

Within this formulation fall many of the traditional optimization problems. An optimization problem in which the objective function and the equality and inequality constraints are linear functions, is called a *linear program*. An optimization problem in which the objective function is quadratic, while the constraints are linear functions is called a *quadratic program*. Convexity plays an important role in many practical optimization problems.

Definition 1.15.1 (Convex Set) A set $\mathscr{X} \in \mathbb{R}^n$ is called *convex* if, for all $\mathbf{x}, \mathbf{y} \in \mathscr{X}$ and $\theta \in (0, 1)$, the point $(\theta\mathbf{x} + (1 - \theta)\mathbf{y}) \in \mathscr{X}$.

Definition 1.15.2 (Convex Function) A function $f(\mathbf{x})$ on a convex set $\mathscr{X}$ is called *convex* if, for all $\mathbf{x}, \mathbf{y} \in \mathscr{X}$ and $\theta \in (0, 1)$,

$$f\big(\theta\mathbf{x} + (1 - \theta)\mathbf{y}\big) \leq \theta f(\mathbf{x}) + (1 - \theta) f(\mathbf{y}) \,. \tag{1.76}$$

If a strict inequality in (1.76) holds, the function is said to be *strictly convex*. If a function f is (strictly) convex, then $-f$ is said to be (strictly) *concave*. Assuming $\mathscr{X}$ is an open set, convexity for $f \in C^1$ is equivalent to

$$f(\mathbf{y}) \geqslant f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T \nabla f(\mathbf{x}) \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathscr{X}.$$

Moreover, for $f \in C^2$, convexity is equivalent to the Hessian matrix being positive semidefinite for all $\mathbf{x} \in \mathscr{X}$:

$$\mathbf{y}^T \left[\nabla^2 f(\mathbf{x})\right] \mathbf{y} \geqslant 0, \quad \text{for all } \mathbf{y} \in \mathbb{R}^n \,.$$

The problem (1.73) is said to be a *convex programming problem* if

1. the objective function f is convex,
2. the inequality constraint functions $\{g_i(\mathbf{x})\}$ are convex, and
3. the equality constraint functions $\{h_i(\mathbf{x})\}$ are *affine*, that is, of the form $\boldsymbol{a}_i^T\mathbf{x} - b_i$.

Note that the last requirement follows from the fact that an equality constraint $h_i(\mathbf{x}) = 0$ can be viewed as a combination of the inequality constraints $h_i(\mathbf{x}) \leqslant 0$ and $-h_i(\mathbf{x}) \leqslant 0$, so that both h_i and $-h_i$ need to be convex. Both the linear and quadratic programs (with positive definite matrix C) are convex.

1.15.1 Lagrangian Method

The main components of the Lagrangian method are the Lagrange multipliers and the Lagrange function. The method was developed by Lagrange in 1797 for the optimization problem (1.73) with equality constraints (1.74). In 1951 Kuhn and Tucker extended Lagrange's method to inequality constraints.

Definition 1.15.3 (Lagrange Function) Given an optimization problem (1.73) containing only equality constraints $h_i(\mathbf{x}) = 0,\ i = 1, \ldots, m$, the *Lagrange function*, or *Lagrangian*, is defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\beta}) = f(\mathbf{x}) + \sum_i \beta_i\, h_i(\mathbf{x}) \ ,$$

where the coefficients $\{\beta_i\}$ are called the *Lagrange multipliers*.

A necessary condition for a point $\mathbf{x}^*$ to be a local minimizer of $f(\mathbf{x})$ subject to the equality constraints $h_i(\mathbf{x}) = 0,\ i = 1, \ldots, m$, is

$$\begin{aligned} \nabla_{\mathbf{x}}\, \mathcal{L}(\mathbf{x}^*, \boldsymbol{\beta}^*) &= \mathbf{0} \ , \\ \nabla_{\boldsymbol{\beta}}\, \mathcal{L}(\mathbf{x}^*, \boldsymbol{\beta}^*) &= \mathbf{0} \end{aligned}$$

for some value $\boldsymbol{\beta}^*$. The above conditions are also sufficient if $\mathcal{L}(\mathbf{x}, \boldsymbol{\beta}^*)$ is a convex function of $\mathbf{x}$.

■ **EXAMPLE 1.17 Maximum Entropy Distribution**

Let $p = \{p_i, i = 1, \ldots, n\}$ be a probability distribution. Consider the following program, which maximizes the (Shannon) entropy:

$$\begin{aligned} \max_{\mathbf{p}} \quad & -\sum_{i=1}^{n} p_i \ln p_i \\ \text{subject to:} \quad & \sum_{i=1}^{n} p_i = 1 \ . \end{aligned}$$

The Lagrangian is

$$\mathcal{L}(\mathbf{p}, \beta) = \sum_{i=1}^{n} p_i \ln p_i + \beta \left(\sum_{i=1}^{n} p_i - 1 \right)$$

over the domain $\{(\mathbf{p}, \beta) : p_i \geq 0, i = 1, \ldots, n,\ \beta \in \mathbb{R}\}$. The optimal solution $\mathbf{p}^*$ of the problem is the uniform distribution, that is, $\mathbf{p}^* = (1/n, \ldots, 1/n)$; see Problem 1.35.

Definition 1.15.4 (Generalized Lagrange Function) Given the original optimization problem (1.73), containing both the equality and inequality constraints, the *generalized Lagrange function*, or simply *Lagrangian*, is defined as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{x}) + \sum_{i=1}^{k} \alpha_i\, g_i(\mathbf{x}) + \sum_{i=1}^{m} \beta_i\, h_i(\mathbf{x}) \ .$$

A necessary condition for a point $\mathbf{x}^*$ to be a local minimizer of $f(\mathbf{x})$ in the optimization problem (1.73) is the existence of an $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$ such that

$$\begin{aligned}
\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) &= \mathbf{0}, \\
\nabla_{\boldsymbol{\beta}} \mathcal{L}(\mathbf{x}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*) &= \mathbf{0}, \\
g_i(\mathbf{x}^*) &\leqslant 0, \quad i = 1, \ldots, k, \\
\alpha_i^* &\geqslant 0, \quad i = 1, \ldots, k, \\
\alpha_i^* \, g_i(\mathbf{x}^*) &= 0, \quad i = 1, \ldots, k.
\end{aligned}$$

These equations are usually referred as the *Karush-Kuhn-Tucker (KKT) conditions*. For *convex* programs we have the following important results:

1. Every local solution $\mathbf{x}^*$ to a convex programming problem is a global solution and the set of global solutions is convex. If, in addition, the objective function is strictly convex, then any global solution is unique.

2. For a strictly convex programming problem with C^1 objective and constraint functions, the KKT conditions are necessary and sufficient for a unique global solution.

1.15.2 Duality

The aim of duality is to provide an alternative formulation of an optimization problem that is often more computationally efficient or has some theoretical significance (see [8], page 219). The original problem (1.73) is referred to as the *primal* problem, whereas the reformulated problem, based on Lagrange multipliers, is referred to as the *dual* problem. Duality theory is most relevant to convex optimization problems. It is well known that if the primal optimization problem is (strictly) convex, then the dual problem is (strictly) concave and has a (unique) solution from which the optimal (unique) primal solution can be deduced.

Definition 1.15.5 (Lagrange Dual Program) The *Lagrange dual program* of the primal program (1.73), is

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \quad \mathcal{L}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\
&\text{subject to:} \quad \boldsymbol{\alpha} \geqslant 0,
\end{aligned}$$

where $\mathcal{L}^*$ is the *Lagrange dual function*:

$$\mathcal{L}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \inf_{\mathbf{x} \in \mathcal{X}} \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta}). \tag{1.77}$$

It is not difficult to see that if f^* is the minimal value of the primal problem, then $\mathcal{L}^*(\boldsymbol{\alpha}, \boldsymbol{\beta}) \leqslant f^*$ for any $\boldsymbol{\alpha} \geqslant 0$ and any $\boldsymbol{\beta}$. This property is called *weak duality*. The Lagrangian dual program thus determines the best lower bound on f^*. If d^* is the optimal value for the dual problem then $d^* < f^*$. The difference $f^* - d^*$ is called the *duality gap*.

The duality gap is extremely useful for providing lower bounds for the solutions of primal problems that may be impossible to solve directly. It is important to note that for linearly constrained problems, if the primal is infeasible (does not have a solution satisfying the constraints), then the dual is either infeasible or unbounded. Conversely, if the dual is infeasible, then the primal has no solution. Of crucial importance is the *strong duality*

theorem, which states that for convex programs (1.73) with linear constrained functions h_i and g_i the duality gap is zero, and any $\mathbf{x}^*$ and $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ satisfying the KKT conditions are (global) solutions to the primal and dual programs, respectively. In particular, this holds for linear and convex quadratic programs (note that not all quadratic programs are convex).

For a convex primal program with C^1 objective and constraint functions, the Lagrangian dual function (1.77) can be obtained by simply setting the gradient (with respect to $\mathbf{x}$) of the Lagrangian $\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ to zero. One can further simplify the dual program by substituting into the Lagrangian the relations between the variables thus obtained.

■ EXAMPLE 1.18 Linear Programming Problem

Consider the following linear programming problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to:} \quad & A\mathbf{x} \geqslant \mathbf{b} \,. \end{aligned}$$

The Lagrangian is $\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = \mathbf{c}^T\mathbf{x} - \boldsymbol{\alpha}^T(A\mathbf{x} - \mathbf{b})$. The Lagrange dual function is the infimum of $\mathcal{L}$ over all $\mathbf{x}$; thus,

$$\mathcal{L}^*(\boldsymbol{\alpha}) = \begin{cases} \mathbf{b}^T\boldsymbol{\alpha} & \text{if } A^T\boldsymbol{\alpha} = \mathbf{c} \,, \\ -\infty & \text{otherwise,} \end{cases}$$

so that the Lagrange dual program becomes

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \mathbf{b}^T\boldsymbol{\alpha} \\ \text{subject to:} \quad & A^T\boldsymbol{\alpha} = \mathbf{c} \\ & \boldsymbol{\alpha} \geqslant \mathbf{0} \,. \end{aligned}$$

It is interesting to note that for the linear programming problem the dual of the dual problem always gives back the primal problem.

■ EXAMPLE 1.19 Quadratic Programming Problem

Consider the following quadratic programming problem:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^T C\mathbf{x} \\ \text{subject to:} \quad & C\mathbf{x} \geqslant \mathbf{b} \,, \end{aligned}$$

where the $n \times n$ matrix C is assumed to be positive definite (for a general quadratic programming problem the matrix C can always be assumed to be symmetric, but it is not necessarily positive definite). The Lagrangian is $\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{x}^T C\mathbf{x} - \boldsymbol{\alpha}^T(C\mathbf{x} - \mathbf{b})$. We can minimize this by taking its gradient with respect to $\mathbf{x}$ and setting it to zero. This gives $C\mathbf{x} - C\boldsymbol{\alpha} = C(\mathbf{x} - \boldsymbol{\alpha}) = \mathbf{0}$. The positive definiteness of C implies that $\mathbf{x} = \boldsymbol{\alpha}$. The maximization of the Lagrangian is now reduced to maximizing $\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^T C\boldsymbol{\alpha} - \boldsymbol{\alpha}^T(C\boldsymbol{\alpha} - \mathbf{b}) = -\frac{1}{2}\boldsymbol{\alpha}^T C\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{b}$ subject to $\boldsymbol{\alpha} \geqslant \mathbf{0}$. Hence, we can write the dual problem as

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\boldsymbol{\alpha}^T C\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\mathbf{b} \\ \text{subject to:} \quad & \boldsymbol{\alpha} \geqslant \mathbf{0} \,. \end{aligned}$$

Notice that the dual problem involves only simple nonnegativity constraints.

Now suppose that we are given the Cholesky factorization $C = BB^T$. It turns out (see Problem 1.36) that the Lagrange dual of the above dual problem can be written as

$$\begin{aligned} \min_{\boldsymbol{\mu}} \quad & \frac{1}{2}\boldsymbol{\mu}^T\boldsymbol{\mu} \\ \text{subject to:} \quad & B\boldsymbol{\mu} \geqslant \mathbf{b}\,, \end{aligned} \tag{1.78}$$

with $\boldsymbol{\mu} = B^T\boldsymbol{\alpha}$. This is a so-called *least distance* problem, which, provided we know the Cholesky factorization of C, is easier to solve than the original quadratic programming problem.

A final example of duality is provided by the widely used *minimum cross-entropy method* [10].

■ **EXAMPLE 1.20 Minimum Cross-Entropy (MinxEnt) Method**

Let $\mathbf{X}$ be a discrete random variable (or vector) taking values $\mathbf{x}_1, \ldots, \mathbf{x}_r$, and let $\mathbf{q} = (q_1, \ldots, q_r)^T$ and $\mathbf{p} = (p_1, \ldots, p_r)^T$ be two strictly positive distribution (column) vectors for $\mathbf{X}$. Consider the following primal program of minimizing the cross-entropy of $\mathbf{p}$ and $\mathbf{q}$, that is, $\sum_{i=1}^n p_i \ln(p_i/q_i)$, for a fixed $\mathbf{q}$, subject to linear equality constraints:

$$\min_{\mathbf{p}} \quad \sum_{k=1}^r p_k \ln \frac{p_k}{q_k} \tag{1.79}$$

$$\text{subject to:} \quad \mathbb{E}_{\mathbf{p}}[S_i(\mathbf{X})] = \sum_{k=1}^r S_i(\mathbf{x}_k)\, p_k = \gamma_i, \quad i = 1, \ldots, m \tag{1.80}$$

$$\sum_{k=1}^r p_k = 1\,, \tag{1.81}$$

where $S_1, \ldots, S_m$ are arbitrary functions.

Here the objective function is convex, since it is a linear combination of functions of the form $p \ln(p/c)$, which are convex on $\mathbb{R}_+$, for any $c > 0$. In addition, the equality constraint functions are affine (of the form $\mathbf{a}^T\mathbf{p} - \gamma$). Therefore, this problem is convex. To derive the optimal solution $\mathbf{p}^*$ of the above primal program, it is typically easier to solve the associated *dual* program [10]. Below we present the corresponding procedure.

1. The Lagrangian of the primal problem is given by

$$\mathcal{L}(\mathbf{p}, \boldsymbol{\lambda}, \beta) = \sum_{k=1}^r p_k \ln \frac{p_k}{q_k} - \sum_{i=1}^m \lambda_i \Big(\sum_{k=1}^r S_i(\mathbf{x}_k)\, p_k - \gamma_i \Big) + \beta \Big(\sum_{k=1}^r p_k - 1 \Big)\,, \tag{1.82}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)^T$ is the Lagrange multiplier vector corresponding to (1.80) and β is the Lagrange multiplier corresponding to (1.81). Note that we can use either a plus or a minus sign in the second sum of (1.82). We choose the latter, because later on we generalize the above problem to inequality ($\geqslant$) constraints in (1.80), giving rise to a minus sign in the Lagrangian.

2. Solve (for fixed $\boldsymbol{\lambda}$ and β)

$$\min_{\mathbf{p}} \mathcal{L}(\mathbf{p}, \boldsymbol{\lambda}, \beta) \tag{1.83}$$

by solving

$$\nabla_{\mathbf{p}} \mathcal{L}(\mathbf{p}, \boldsymbol{\lambda}, \beta) = \mathbf{0} ,$$

which gives the set of equations

$$\nabla_{p_k} \mathcal{L}(\mathbf{p}, \boldsymbol{\lambda}, \beta) = \ln \frac{p_k}{q_k} + 1 - \sum_{i=1}^{m} \lambda_i S_i(\mathbf{x}_k) + \beta = 0, \quad k = 1, \ldots, r .$$

Denote the optimal solution and the optimal function value obtained from the program (1.83) as $\mathbf{p}(\boldsymbol{\lambda}, \beta)$ and $\mathcal{L}^*(\boldsymbol{\lambda}, \beta)$, respectively. The latter is the Lagrange dual function. We have

$$p_k(\boldsymbol{\lambda}, \beta) = q_k \exp\left(-\beta - 1 + \sum_{i=1}^{m} \lambda_i S_i(\mathbf{x}_k)\right), \quad k = 1, \ldots, r . \tag{1.84}$$

Since the sum of the $\{p_k\}$ must be 1, we obtain

$$\mathrm{e}^{\beta} = \sum_{k=1}^{r} q_k \exp\left(-1 + \sum_{i=1}^{m} \lambda_i S_i(\mathbf{x}_k)\right) . \tag{1.85}$$

Substituting $\mathbf{p}(\boldsymbol{\lambda}, \beta)$ back into the Lagrangian gives

$$\mathcal{L}^*(\boldsymbol{\lambda}, \beta) = -1 + \sum_{i=1}^{m} \lambda_i \gamma_i - \beta . \tag{1.86}$$

3. Solve the *dual* program

$$\max_{\boldsymbol{\lambda}, \beta} \mathcal{L}^*(\boldsymbol{\lambda}, \beta) . \tag{1.87}$$

Since β and $\boldsymbol{\lambda}$ are related via (1.85), solving (1.87) can be done by substituting the corresponding $\beta(\boldsymbol{\lambda})$ into (1.86) and optimizing the resulting function:

$$D(\boldsymbol{\lambda}) = -1 + \sum_{i=1}^{m} \lambda_i \gamma_i - \ln\left\{\sum_{k=1}^{m} q_k \exp\{-1 + \textstyle\sum_{i=1}^{m} \lambda_i S_i(\mathbf{x}_k)\}\right\} . \tag{1.88}$$

Since $D(\boldsymbol{\lambda})$ is continuously differentiable and concave with respect to $\boldsymbol{\lambda}$, we can derive the optimal solution, $\boldsymbol{\lambda}^*$, by solving

$$\nabla_{\boldsymbol{\lambda}} D(\boldsymbol{\lambda}) = \mathbf{0} , \tag{1.89}$$

which can be written componentwise in the following explicit form:

$$\begin{aligned} \nabla_{\lambda_j} D(\boldsymbol{\lambda}) &= \gamma_i - \frac{\sum_{k=1}^{r} S_i(\mathbf{x}_k)\, q_k \exp\left\{-1 + \sum_{j=1}^{m} \lambda_j S_j(\mathbf{x}_k)\right\}}{\sum_{k=1}^{r} q_k \exp\left\{-1 + \sum_{j=1}^{m} \lambda_j S_j(\mathbf{x}_k)\right\}} \\ &= \gamma_i - \frac{\mathbb{E}_{\mathbf{q}}\left[S_i(\mathbf{X}) \exp\left\{-1 + \sum_{j=1}^{m} \lambda_j S_j(\mathbf{X})\right\}\right]}{\mathbb{E}_{\mathbf{q}}\left[\exp\left\{-1 + \sum_{j=1}^{m} \lambda_j S_j(\mathbf{X})\right\}\right]} = 0 \end{aligned} \tag{1.90}$$

for $j = 1, \ldots, m$. The optimal vector $\boldsymbol{\lambda}^* = (\lambda_1^*, \ldots, \lambda_m^*)$ can be found by solving (1.90) numerically. Note that if the primal program has a nonempty interior optimal solution, then the dual program has an optimal solution $\boldsymbol{\lambda}^*$.

4. Finally, substitute $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$ and $\beta = \beta(\boldsymbol{\lambda}^*)$ back into (1.84) to obtain the solution to the original MinxEnt program.

It is important to note that we do not need to explicitly impose the conditions $p_i \geqslant 0$, $i = 1, \ldots, n$, because the quantities $\{p_i\}$ in (1.84) are automatically strictly positive. This is a crucial property of the CE distance; see also [2]. It is instructive (see Problem 1.37) to verify how adding the nonnegativity constraints affects the above procedure.

When inequality constraints $\mathbb{E}_{\mathbf{p}}[S_i(\mathbf{X})] \geqslant \gamma_i$ are used in (1.80) instead of equality constraints, the solution procedure remains almost the same. The only difference is that the Lagrange multiplier vector $\boldsymbol{\lambda}$ must now be nonnegative. It follows that the dual program becomes

$$\begin{aligned} &\max_{\boldsymbol{\lambda}} \quad D(\boldsymbol{\lambda}) \\ &\text{subject to:} \quad \boldsymbol{\lambda} \geqslant \mathbf{0}\,, \end{aligned}$$

with $D(\boldsymbol{\lambda})$ given in (1.88).

A further generalization is to replace the above discrete optimization problem with a *functional* optimization problem. This topic will be discussed in Chapter 9. In particular, Section 9.5 deals with the MinxEnt method, which involves a functional MinxEnt problem.

PROBLEMS

Probability Theory

1.1 Prove the following results, using the properties of the probability measure in Definition 1.1.1 (here A and B are events):

a) $\mathbb{P}(A^c) = 1 - \mathbb{P}(A)$.

b) $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$.

1.2 Prove the product rule (1.4) for the case of three events.

1.3 We draw three balls consecutively from a bowl containing exactly five white and five black balls, without putting them back. What is the probability that all drawn balls will be black?

1.4 Consider the random experiment where we toss a biased coin until heads comes up. Suppose the probability of heads on any one toss is p. Let X be the number of tosses required. Show that $X \sim \mathsf{G}(p)$.

1.5 In a room with many people, we ask each person his/her birthday, for example May 5. Let N be the number of people queried until we get a "duplicate" birthday.

a) Calculate $\mathbb{P}(N > n)$, $n = 0, 1, 2, \ldots$.

b) For which n do we have $\mathbb{P}(N \leqslant n) \geqslant 1/2$?

c) Use a computer to calculate $\mathbb{E}[N]$.

1.6 Let X and Y be independent standard normal random variables, and let U and V be random variables that are derived from X and Y via the linear transformation

$$\begin{pmatrix} U \\ V \end{pmatrix} = \begin{pmatrix} \sin\alpha & -\cos\alpha \\ \cos\alpha & \sin\alpha \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} .$$

a) Derive the joint pdf of U and V.

b) Show that U and V are independent and standard normally distributed.

1.7 Let $X \sim \mathsf{Exp}(\lambda)$. Show that the *memoryless property* holds: for all $s, t \geqslant 0$,

$$\mathbb{P}(X > t + s \mid X > t) = \mathbb{P}(X > s) .$$

1.8 Let X_1, X_2, X_3 be independent Bernoulli random variables with success probabilities $1/2, 1/3$, and $1/4$, respectively. Give their conditional joint pdf, given that $X_1 + X_2 + X_3 = 2$.

1.9 Verify the expectations and variances in Table 1.3.

1.10 Let X and Y have joint density f given by

$$f(x, y) = c\,x\,y, \quad 0 \leqslant y \leqslant x, \quad 0 \leqslant x \leqslant 1 .$$

a) Determine the normalization constant c.

b) Determine $\mathbb{P}(X + 2\,Y \leqslant 1)$.

1.11 Let $X \sim \mathsf{Exp}(\lambda)$ and $Y \sim \mathsf{Exp}(\mu)$ be independent. Show that

a) $\min(X, Y) \sim \mathsf{Exp}(\lambda + \mu)$,

b) $\mathbb{P}(X < Y \mid \min(X, Y)) = \dfrac{\lambda}{\lambda + \mu}$.

1.12 Verify the properties of variance and covariance in Table 1.4.

1.13 Show that the correlation coefficient always lies between -1 and 1. [Hint, use the fact that the variance of $aX + Y$ is always non-negative, for any a.]

1.14 Consider Examples 1.1 and 1.2. Define X as the function that assigns the number $x_1 + \cdots + x_n$ to each outcome $\omega = (x_1, \ldots, x_n)$. The event that there are exactly k heads in n throws can be written as

$$\{\omega \in \Omega : X(\omega) = k\} .$$

If we abbreviate this to $\{X = k\}$, and further abbreviate $\mathbb{P}(\{X = k\})$ to $\mathbb{P}(X = k)$, then we obtain exactly (1.7). Verify that one can always view random variables in this way, that is, as real-valued functions on Ω, and that probabilities such as $\mathbb{P}(X \leqslant x)$ should be interpreted as $\mathbb{P}(\{\omega \in \Omega : X(\omega) \leqslant x\})$.

1.15 Show that

$$\mathrm{Var}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i=1}^{n} \mathrm{Var}(X_i) + 2 \sum_{i<j} \mathrm{Cov}(X_i, X_j) .$$

1.16 Let Σ be the covariance matrix of a random column vector $\mathbf{X}$. Write $\mathbf{Y} = \mathbf{X} - \boldsymbol{\mu}$, where $\boldsymbol{\mu}$ is the expectation vector of $\mathbf{X}$. Hence, $\Sigma = \mathbb{E}[\mathbf{Y}\mathbf{Y}^T]$. Show that Σ is positive semidefinite. That is, for any vector $\mathbf{u}$, we have $\mathbf{u}^T \Sigma \mathbf{u} \geqslant 0$.

1.17 Suppose $Y \sim \mathsf{Gamma}(n, \lambda)$. Show that for all $x \geqslant 0$

$$\mathbb{P}(Y \leqslant x) = 1 - \sum_{k=0}^{n-1} \frac{\mathrm{e}^{-\lambda x}(\lambda x)^k}{k!} \,. \tag{1.91}$$

1.18 Consider the random experiment where we draw uniformly and independently n numbers, $X_1, \ldots, X_n$, from the interval [0,1].

a) Let M be the smallest of the n numbers. Express M in terms of $X_1, \ldots, X_n$.

b) Determine the pdf of M.

1.19 Let $Y = \mathrm{e}^X$, where $X \sim \mathsf{N}(0, 1)$.

a) Determine the pdf of Y.

b) Determine the expected value of Y.

1.20 We select a point (X, Y) from the triangle $(0, 0) - (1, 0) - (1, 1)$ in such a way that X has a uniform distribution on $(0, 1)$ and the conditional distribution of Y given $X = x$ is uniform on $(0, x)$.

a) Determine the joint pdf of X and Y.

b) Determine the pdf of Y.

c) Determine the conditional pdf of X given $Y = y$ for all $y \in (0, 1)$.

d) Calculate $\mathbb{E}[X \mid Y = y]$ for all $y \in (0, 1)$.

e) Determine the expectations of X and Y.

Poisson Processes

1.21 Let $\{N_t, t \geqslant 0\}$ be a Poisson process with rate $\lambda = 2$. Find

a) $\mathbb{P}(N_2 = 1, N_3 = 4, N_5 = 5)$,

b) $\mathbb{P}(N_4 = 3 \mid N_2 = 1, N_3 = 2)$,

c) $\mathbb{E}[N_4 \mid N_2 = 2]$,

d) $\mathbb{P}(N[2, 7] = 4, N[3, 8] = 6)$,

e) $\mathbb{E}[N[4, 6] \mid N[1, 5] = 3]$.

1.22 Show that for any fixed $k \in \mathbb{N}$, $t > 0$ and $\lambda > 0$,

$$\lim_{n\to\infty} \binom{n}{k} \left(\frac{\lambda t}{n}\right)^k \left(1 - \frac{\lambda t}{n}\right)^{n-k} = \frac{(\lambda t)^k}{k!} \mathrm{e}^{-\lambda t} \,.$$

(Hint: write out the binomial coefficient and use the fact that $\lim_{n\to\infty} \left(1 - \frac{\lambda t}{n}\right)^n = \mathrm{e}^{-\lambda t}$.)

1.23 Consider the Bernoulli approximation in Section 1.11. Let $U_1, U_2, \ldots$ denote the times of success for the Bernoulli process X.

a) Verify that the "intersuccess" times $U_1, U_2 - U_1, \ldots$ are independent and have a geometric distribution with parameter $p = \lambda h$.

b) For small h and $n = \lfloor t/h \rfloor$, show that the relationship $\mathbb{P}(A_1 > t) \approx \mathbb{P}(U_1 > n)$ leads in the limit, as $n \to \infty$, to

$$\mathbb{P}(A_1 > t) = \mathrm{e}^{-\lambda t}.$$

1.24 If $\{N_t, t \geqslant 0\}$ is a Poisson process with rate λ, show that for $0 \leqslant u \leqslant t$ and $j = 0, 1, 2, \ldots, n$,

$$\mathbb{P}(N_u = j \mid N_t = n) = \binom{n}{j} \left(\frac{u}{t}\right)^j \left(1 - \frac{u}{t}\right)^{n-j},$$

that is, the conditional distribution of N_u given $N_t = n$ is binomial with parameters n and u/t.

Markov Processes

1.25 Determine the (discrete) pdf of each X_n, $n = 0, 1, 2, \ldots$ for the random walk in Example 1.10. Also, calculate $\mathbb{E}[X_n]$ and the variance of X_n for each n.

1.26 Let $\{X_n, n \in \mathbb{N}\}$ be a Markov chain with state space $\{0, 1, 2\}$, transition matrix

$$P = \begin{pmatrix} 0.3 & 0.1 & 0.6 \\ 0.4 & 0.4 & 0.2 \\ 0.1 & 0.7 & 0.2 \end{pmatrix},$$

and initial distribution $\pi = (0.2, 0.5, 0.3)$. Determine

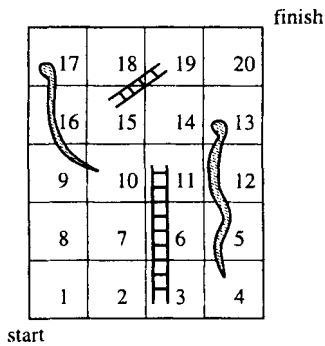
a) $\mathbb{P}(X_1 = 2)$,
b) $\mathbb{P}(X_2 = 2)$,
c) $\mathbb{P}(X_3 = 2 \mid X_0 = 0)$,
d) $\mathbb{P}(X_0 = 1 \mid X_1 = 2)$,
e) $\mathbb{P}(X_1 = 1, X_3 = 1)$.

1.27 Consider two dogs harboring a total number of m fleas. Spot initially has b fleas and Lassie has the remaining $m - b$. The fleas have agreed on the following immigration policy: at every time $n = 1, 2 \ldots$ a flea is selected at random from the total population and that flea will jump from one dog to the other. Describe the flea population on Spot as a Markov chain and find its stationary distribution.

1.28 Classify the states of the Markov chain with the following transition matrix:

$$P = \begin{pmatrix} 0.0 & 0.3 & 0.6 & 0.0 & 0.1 \\ 0.0 & 0.3 & 0.0 & 0.7 & 0.0 \\ 0.3 & 0.1 & 0.6 & 0.0 & 0.0 \\ 0.0 & 0.1 & 0.0 & 0.9 & 0.0 \\ 0.1 & 0.1 & 0.2 & 0.0 & 0.6 \end{pmatrix}.$$

1.29 Consider the following snakes-and-ladders game. Let N be the number of tosses required to reach the finish using a fair die. Calculate the expectation of N using a computer.



1.30 Ms. Ella Brum walks back and forth between her home and her office every day. She owns three umbrellas, which are distributed over two umbrella stands (one at home and one at work). When it is not raining, Ms. Brum walks without an umbrella. When it is raining, she takes one umbrella from the stand at the place of her departure, provided there is one available. Suppose the probability that it is raining at the time of any departure is p. Let X_n denote the number of umbrellas available at the place where Ella arrives after walk number n; $n = 1, 2, \ldots$, including the one that she possibly brings with her. Calculate the limiting probability that it rains and no umbrella is available.

1.31 A mouse is let loose in the maze of Figure 1.9. From each compartment the mouse chooses one of the adjacent compartments with equal probability, independent of the past. The mouse spends an exponentially distributed amount of time in each compartment. The mean time spent in each of the compartments 1, 3, and 4 is two seconds; the mean time spent in compartments 2, 5, and 6 is four seconds. Let $\{X_t, t \geqslant 0\}$ be the Markov jump process that describes the position of the mouse for times $t \geqslant 0$. Assume that the mouse starts in compartment 1 at time $t = 0$.

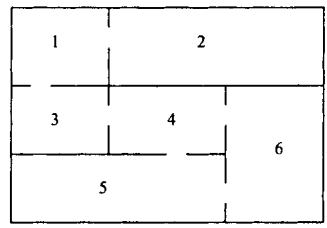


Figure 1.9 A maze.

What are the probabilities that the mouse will be found in each of the compartments $1, 2, \ldots, 6$ at some time t far away in the future?

1.32 In an $M/M/\infty$-queueing system, customers arrive according to a Poisson process with rate a. Every customer who enters is immediately served by one of an infinite number of servers; hence, there is no queue. The service times are exponentially distributed, with mean $1/b$. All service and interarrival times are independent. Let X_t be the number of customers in the system at time t. Show that the limiting distribution of X_t, as $t \to \infty$, is Poisson with parameter a/b.

Optimization

1.33 Let $\mathbf{a}$ and let $\mathbf{x}$ be n-dimensional column vectors. Show that $\nabla_{\mathbf{x}}\, \mathbf{a}^T\mathbf{x} = \mathbf{a}$.

1.34 Let A be a symmetric $n \times n$ matrix and $\mathbf{x}$ be an n-dimensional column vector. Show that $\nabla_{\mathbf{x}}\, \frac{1}{2}\mathbf{x}^T A\mathbf{x} = A\mathbf{x}$. What is the gradient if A is not symmetric?

1.35 Show that the optimal distribution $\mathbf{p}^*$ in Example 1.17 is given by the uniform distribution.

1.36 Derive the program (1.78).

1.37 Consider the MinxEnt program

$$\min_{\mathbf{p}} \sum_{i=1}^{n} p_i \ln \frac{p_i}{q_i}$$

$$\text{subject to: } \mathbf{p} \geqslant \mathbf{0}, \quad A\mathbf{p} = \mathbf{b}, \quad \sum_{i=1}^{n} p_i = 1 \,,$$

where $\mathbf{p}$ and $\mathbf{q}$ are probability distribution vectors and A is an $m \times n$ matrix.

a) Show that the Lagrangian for this problem is of the form

$$\mathcal{L}(\mathbf{p}, \boldsymbol{\lambda}, \beta, \boldsymbol{\mu}) = \mathbf{p}^T \boldsymbol{\xi}(\mathbf{p}) - \boldsymbol{\lambda}^T (A\mathbf{p} - \mathbf{b}) - \boldsymbol{\mu}^T \mathbf{p} + \beta(\mathbf{1}^T \mathbf{p} - 1) \,.$$

b) Show that $p_i = q_i \exp(-\beta - 1 + \mu_i + \sum_{j=1}^{m} \lambda_j a_{ji})$, for $i = 1, \ldots, n$.

c) Explain why, as a result of the KKT conditions, the optimal $\boldsymbol{\mu}^*$ must be equal to the zero vector.

d) Show that the solution to this MinxEnt program is exactly the same as for the program where the nonnegativity constraints are omitted.

Further Reading

An easy introduction to probability theory with many examples is [14], and a more detailed textbook is [9]. A classical reference is [7]. An accurate and accessible treatment of various stochastic processes is given in [4]. For convex optimization we refer to [3] and [8].

REFERENCES

1. S. Asmussen and R. Y. Rubinstein. Complexity properties of steady-state rare-events simulation in queueing models. In J. H. Dshalalow, editor, *Advances in Queueing: Theory, Methods and Open Problems*, pages 429–462, New York, 1995. CRC Press.
2. Z. I. Botev, D. P. Kroese, and T. Taimre. Generalized cross-entropy methods for rare-event simulation and optimization. *Simulation: Transactions of the Society for Modeling and Simulation International*, 2007. In press.
3. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2004.
4. E. Çinlar. *Introduction to Stochastic Processes*. Prentice Hall, Englewood Cliffs, NJ, 1975.
5. T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.
6. C. W. Curtis. *Linear Algebra: An Introductory Approach*. Springer-Verlag, New York, 1984.
7. W. Feller. *An Introduction to Probability Theory and Its Applications*, volume I. John Wiley & Sons, New York, 2nd edition, 1970.
8. R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, 1987.
9. G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Oxford University Press, Oxford, 3rd edition, 2001.
10. J. N. Kapur and H. K. Kesavan. *Entropy Optimization Principles with Applications*. Academic Press, New York, 1992.

11. A. I. Khinchin. *Information Theory*. Dover Publications, New York, 1957.

12. V. Kriman and R. Y. Rurbinstein. Polynomial time algorithms for estimation of rare events in queueing models. In J. Dshalalow, editor, *Frontiers in Queueing: Models and Applications in Science and Engineering*, pages 421–448, New York, 1995. CRC Press.

13. E. L. Lehmann. *Testing Statistical Hypotheses*. Springer-Verlag, New York, 1997.

14. S. M. Ross. *A First Course in Probability*. Prentice Hall, Englewood Cliffs, NJ, 7th edition, 2005.

15. R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling*. John Wiley & Sons, New York, 1998.

This Page Intentionally Left Blank

CHAPTER 2

RANDOM NUMBER, RANDOM VARIABLE, AND STOCHASTIC PROCESS GENERATION

2.1 INTRODUCTION

This chapter deals with the computer generation of random numbers, random variables, and stochastic processes. In a typical stochastic simulation, randomness is introduced into simulation models via independent uniformly distributed random variables. These random variables are then used as building blocks to simulate more general stochastic systems.

The rest of this chapter is organized as follows. We start, in Section 2.2, with the generation of uniform random variables. Section 2.3 discusses general methods for generating one-dimensional random variables. Section 2.4 presents specific algorithms for generating variables from commonly used continuous and discrete distributions. In Section 2.5 we discuss the generation of random vectors. Sections 2.6 and 2.7 treat the generation of Poisson processes, Markov chains and Markov jump processes. Finally, Section 2.8 deals with the generation of random permutations.

2.2 RANDOM NUMBER GENERATION

In the early days of simulation, randomness was generated by *manual* techniques, such as coin flipping, dice rolling, card shuffling, and roulette spinning. Later on, *physical devices*, such as noise diodes and Geiger counters, were attached to computers for the same purpose. The prevailing belief held that only mechanical or electronic devices could produce truly random sequences. Although mechanical devices are still widely used in gambling

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

and lotteries, these methods were abandoned by the computer-simulation community for several reasons: (a) Mechanical methods were too slow for general use, (b) the generated sequences cannot be reproduced and, (c) it has been found that the generated numbers exhibit both bias and dependence. Although certain modern physical generation methods are fast and would pass most statistical tests for randomness (for example, those based on the universal background radiation or the noise of a PC chip), their main drawback remains their lack of repeatability. Most of today's random number generators are not based on physical devices, but on simple algorithms that can be easily implemented on a computer. They are fast, require little storage space, and can readily reproduce a given sequence of random numbers. Importantly, a good random number generator captures all the important statistical properties of true random sequences, even though the sequence is generated by a deterministic algorithm. For this reason, these generators are sometimes called *pseudorandom*.

The most common methods for generating pseudorandom sequences use the so-called *linear congruential generators*, introduced in [6]. These generate a deterministic sequence of numbers by means of the recursive formula

$$X_{i+1} = aX_i + c \pmod{m}, \tag{2.1}$$

where the initial value, X_0, is called the *seed* and the a, c, and m (all positive integers) are called the *multiplier*, the *increment* and the *modulus*, respectively. Note that applying the modulo-m operator in (2.1) means that $aX_i + c$ is divided by m, and the remainder is taken as the value for X_{i+1}. Thus, each X_i can only assume a value from the set $\{0, 1, \ldots, m-1\}$, and the quantities

$$U_i = \frac{X_i}{m}, \tag{2.2}$$

called *pseudorandom numbers*, constitute approximations to a true sequence of uniform random variables. Note that the sequence $X_0, X_1, X_2, \ldots$ will repeat itself after at most m steps and will therefore be periodic, with period not exceeding m. For example, let $a = c = X_0 = 3$ and $m = 5$. Then the sequence obtained from the recursive formula $X_{i+1} = 3X_i + 3 \pmod 5$ is $3, 2, 4, 0, 3$, which has period 4. In the special case where $c = 0$, (2.1) simply reduces to

$$X_{i+1} = a\,X_i \pmod{m}. \tag{2.3}$$

Such a generator is called a *multiplicative congruential generator*. It is readily seen that an arbitrary choice of X_0, a, c, and m will not lead to a pseudorandom sequence with good statistical properties. In fact, number theory has been used to show that only a few combinations of these produce satisfactory results. In computer implementations, m is selected as a large prime number that can be accommodated by the computer word size. For example, in a binary 32-bit word computer, statistically acceptable generators can be obtained by choosing $m = 2^{31} - 1$ and $a = 7^5$, provided that the first bit is a sign bit. A 64-bit or 128-bit word computer will naturally yield better statistical results.

Formulas (2.1), (2.2), and (2.3) can be readily extended to pseudorandom vector generation. For example, the n-dimensional versions of (2.3) and (2.2) can be written as

$$\mathbf{X}_{i+1} = A\mathbf{X}_i \pmod{\mathbf{M}} \tag{2.4}$$

and

$$\mathbf{U}_i = \mathbf{M}^{-1}\mathbf{X}_i, \tag{2.5}$$

respectively, where A is a nonsingular $n \times n$ matrix, $\mathbf{M}$ and $\mathbf{X}$ are n-dimensional vectors, and $\mathbf{M}^{-1}\mathbf{X}_i$ is the n-dimensional vector with components $M_1^{-1}X_1, \ldots, M_n^{-1}X_n$.

Besides the linear congruential generators, other classes have been proposed to achieve longer periods and better statistical properties (see [5]).

Most computer languages already contain a built-in pseudorandom number generator. The user is typically requested only to input the initial seed, X_0, and upon invocation the random number generator produces a sequence of independent, uniform $(0, 1)$ random variables. We therefore assume in this book the availability of such a "black box" that is capable of producing a stream of pseudorandom numbers. In Matlab, for example, this is provided by the `rand` function.

■ **EXAMPLE 2.1 Generating Uniform Random Variables in Matlab**

This example illustrates the use of the `rand` function in Matlab to generate samples from the U(0, 1) distribution. For clarity we have omitted the "`ans =`" output in the Matlab session below.

```
>> rand                    % generate a uniform random number
    0.0196
>> rand                    % generate another uniform random number
    0.823
>> rand(1,4)               % generate a uniform random vector
    0.5252  0.2026  0.6721  0.8381
rand('state',1234)         % set the seed to 1234
>> rand                    % generate a uniform random number
    0.6104
rand('state',1234)         % reset the seed to 1234
>> rand
    0.6104                 % the previous outcome is repeated
```

2.3 RANDOM VARIABLE GENERATION

In this section we discuss various general methods for generating one-dimensional random variables from a prescribed distribution. We consider the inverse-transform method, the alias method, the composition method, and the acceptance-rejection method.

2.3.1 Inverse-Transform Method

Let X be a random variable with cdf F. Since F is a nondecreasing function, the inverse function F^{-1} may be defined as

$$F^{-1}(y) = \inf\{x : F(x) \geqslant y\}\,, \quad 0 \leqslant y \leqslant 1\,. \tag{2.6}$$

(Readers not acquainted with the notion inf should read min.) It is easy to show that if $U \sim \mathrm{U}(0, 1)$, then

$$X = F^{-1}(U) \tag{2.7}$$

has cdf F. Namely, since F is invertible and $\mathbb{P}(U \leqslant u) = u$, we have

$$\mathbb{P}(X \leqslant x) = \mathbb{P}(F^{-1}(U) \leqslant x) = \mathbb{P}(U \leqslant F(x)) = F(x)\,. \tag{2.8}$$

Thus, to generate a random variable X with cdf F, draw $U \sim \mathsf{U}(0,1)$ and set $X = F^{-1}(U)$. Figure 2.1 illustrates the inverse-transform method given by the following algorithm.

Algorithm 2.3.1 (The Inverse-Transform Method)

1. *Generate U from* $\mathsf{U}(0,1)$.
2. *Return* $X = F^{-1}(U)$.

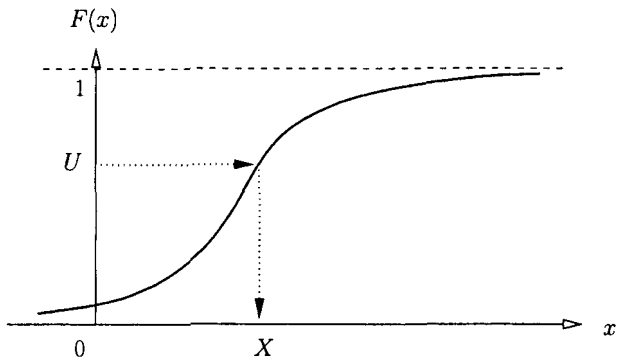


Figure 2.1 Inverse-transform method.

■ **EXAMPLE 2.2**

Generate a random variable from the pdf

$$f(x) = \begin{cases} 2x, & 0 \leqslant x \leqslant 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

The cdf is

$$F(x) = \begin{cases} 0, & x < 0 \\ \int_0^x 2y\,dy = x^2, & 0 \leqslant x \leqslant 1 \\ 1, & x > 1. \end{cases}$$

Applying (2.7), we have

$$X = F^{-1}(U) = \sqrt{U}\ .$$

Therefore, to generate a random variable X from the pdf (2.9), first generate a random variable U from $\mathsf{U}(0,1)$ and then take its square root.

■ **EXAMPLE 2.3 Order Statistics**

Let $X_1, \ldots, X_n$ be iid random variables with cdf F. We wish to generate random variables $X_{(n)}$ and $X_{(1)}$ that are distributed according to the order statistics $\max(X_1, \ldots, X_n)$ and $\min(X_1, \ldots, X_n)$, respectively. From Example 1.7 we see

that the cdfs of $X_{(n)}$ and $X_{(1)}$ are $F_n(x) = [F(x)]^n$ and $F_1(x) = 1 - [1 - F(x)]^n$, respectively. Applying (2.7), we get

$$X_{(n)} = F^{-1}(U^{1/n})$$

and, since $1 - U$ is also from U(0, 1),

$$X_{(1)} = F^{-1}(1 - U^{1/n}) .$$

In the special case where $F(x) = x$, that is, $X_i \sim \mathsf{U}(0, 1)$, we have

$$X_{(n)} = U^{1/n} \quad \text{and} \quad X_{(1)} = 1 - U^{1/n} .$$

■ **EXAMPLE 2.4 Drawing from a Discrete Distribution**

Let X be a discrete random variable with $\mathbb{P}(X = x_i) = p_i$, $i = 1, 2, \ldots$, with $\sum_i p_i = 1$ and $x_1 < x_2 < \ldots$. The cdf F of X is given by $F(x) = \sum_{i:x_i \leqslant x} p_i$, $i = 1, 2, \ldots$ and is illustrated in Figure 2.2.

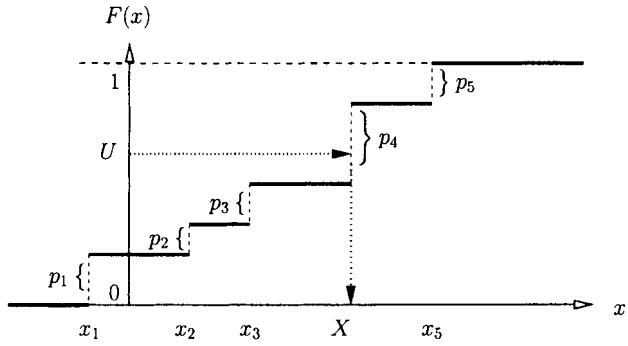


Figure 2.2 Inverse-transform method for a discrete random variable.

Hence, the algorithm for generating a random variable from F can be written as follows.

Algorithm 2.3.2 (Inverse-Transform Method for a Discrete Distribution)

1. *Generate* $U \sim \mathsf{U}(0, 1)$.
2. *Find the smallest positive integer,* k, *such that* $U \leqslant F(x_k)$ *and return* $X = x_k$.

Much of the execution time in Algorithm 2.3.2 is spent in making the comparisons of Step 2. This time can be reduced by using efficient search techniques (see [2]).

In general, the inverse-transform method requires that the underlying cdf, F, exists in a form for which the corresponding inverse function F^{-1} can be found analytically or algorithmically. Applicable distributions are, for example, the exponential, uniform, Weibull, logistic, and Cauchy distributions. Unfortunately, for many other probability distributions, it is either impossible or difficult to find the inverse transform, that is, to solve

$$F(x) = \int_{-\infty}^{x} f(t)\, dt = u$$

with respect to x. Even in the case where F^{-1} exists in an explicit form, the inverse-transform method may not necessarily be the most efficient random variable generation method (see [2]).

2.3.2 Alias Method

An alternative to the inverse-transform method for generating discrete random variables, which does not require time-consuming search techniques as per Step 2 of Algorithm 2.3.2, is the so-called *alias method* [11]. It is based on the fact that an arbitrary discrete n-point pdf f, with

$$f(x_i) = \mathbb{P}(X = x_i), \quad i = 1, \ldots, n \,,$$

can be represented as an equally weighted mixture of $n-1$ pdfs, $q^{(k)}$, $k = 1, \ldots, n-1$, each having at most *two* nonzero components. That is, any n-point pdf f can be represented as

$$f(x) = \frac{1}{n-1} \sum_{k=1}^{n-1} q^{(k)}(x) \tag{2.10}$$

for suitably defined two-point pdfs $q^{(k)}$, $k = 1, \ldots, n-1$; see [11].

The alias method is rather general and efficient but requires an initial setup and extra storage for the $n-1$ pdfs, $q^{(k)}$. A procedure for computing these two-point pdfs can be found in [2]. Once the representation (2.10) has been established, generation from f is simple and can be written as follows:

Algorithm 2.3.3 (Alias Method)

1. *Generate* $U \sim \mathrm{U}(0,1)$ *and set* $K = 1 + \lfloor (n-1)\,U \rfloor$.
2. *Generate* X *from the two-point pdf* $q^{(K)}$.

2.3.3 Composition Method

This method assumes that a cdf, F, can be expressed as a *mixture* of cdfs $\{G_i\}$, that is:

$$F(x) = \sum_{i=1}^{m} p_i \, G_i(x) \,, \tag{2.11}$$

where

$$p_i > 0, \quad \sum_{i=1}^{m} p_i = 1 \,.$$

Let $X_i \sim G_i$ and let Y be a discrete random variable with $\mathbb{P}(Y = i) = p_i$, independent of X_i, for $1 \leqslant i \leqslant m$. Then a random variable X with cdf F can be represented as

$$X = \sum_{i=1}^{m} X_i \, I_{\{Y=i\}} \,.$$

It follows that in order to generate X from F, we must first generate the discrete random variable Y and then, given $Y = i$, generate X_i from G_i. We thus have the following method.

Algorithm 2.3.4 (Composition Method)

1. *Generate the random variable Y according to*

$$\mathbb{P}(Y = i) = p_i, \quad i = 1, \ldots, m .$$

2. *Given $Y = i$, generate X from the cdf G_i.*

2.3.4 Acceptance-Rejection Method

The inverse-transform and composition methods are direct methods in the sense that they deal directly with the cdf of the random variable to be generated. The acceptance-rejection method, is an indirect method due to Stan Ulam and John von Neumann. It can be applied when the above-mentioned direct methods either fail or turn out to be computationally inefficient.

To introduce the idea, suppose that the *target* pdf f (the pdf from which we want to sample) is bounded on some finite interval $[a, b]$ and is zero outside this interval (see Figure 2.3). Let

$$c = \sup\{f(x) : x \in [a, b]\} .$$

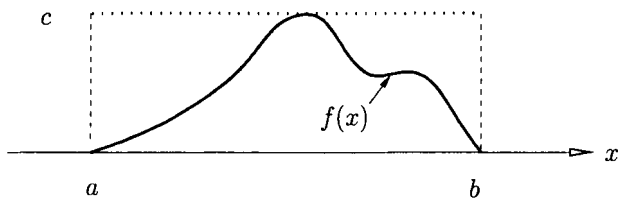


Figure 2.3 The acceptance-rejection method.

In this case, generating a random variable $Z \sim f$ is straightforward, and it can be done using the following acceptance-rejection steps:

1. Generate $X \sim \mathsf{U}(a, b)$.

2. Generate $Y \sim \mathsf{U}(0, c)$ independently of X.

3. If $Y \leqslant f(X)$, return $Z = X$. Otherwise, return to Step 1.

It is important to note that each generated vector (X, Y) is uniformly distributed over the rectangle $[a, b] \times [0, c]$. Therefore, the accepted pair (X, Y) is uniformly distributed under the graph f. This implies that the distribution of the accepted values of X has the desired pdf f.

We can generalize this as follows. Let g be an arbitrary density such that $\phi(x) = C\, g(x)$ *majorizes* $f(x)$ for some constant C (Figure 2.4); that is, $\phi(x) \geqslant f(x)$ for all x. Note that of necessity $C \geqslant 1$. We call $g(x)$ the *proposal* pdf and assume that it is easy to generate random variables from it.

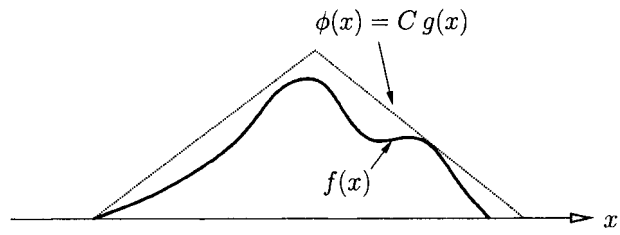


Figure 2.4 The acceptance-rejection method with a majorizing function ϕ.

The acceptance-rejection algorithm can be written as follows:

Algorithm 2.3.5 (Acceptance-Rejection Algorithm)

1. *Generate X from $g(x)$.*
2. *Generate $Y \sim \mathrm{U}(0, C\,g(X))$.*
3. *If $Y \leqslant f(X)$, return $Z = X$. Otherwise, return to Step 1.*

The theoretical basis of the acceptance-rejection method is provided by the following theorem.

Theorem 2.3.1 *The random variable generated according to Algorithm 2.3.5 has the desired pdf $f(x)$.*

Proof: Define the following two subsets:

$$\mathscr{A} = \{(x, y) : 0 \leq y \leq Cg(x)\} \quad \text{and} \quad \mathscr{B} = \{(x, y) : 0 \leq y \leq f(x)\}\,, \tag{2.12}$$

which represent the areas below the curves $Cg(x)$ and $f(x)$, respectively. Note first that Steps 1 and 2 of Algorithm 2.3.5 imply that the random vector (X, Y) is uniformly distributed on $\mathscr{A}$. To see this, let $q(x, y)$ denote the joint pdf of (X, Y), and let $q(y \mid x)$ denote the conditional pdf of Y given $X = x$. Then we have

$$q(x, y) = \begin{cases} g(x)\, q(y \mid x) & \text{if} (x, y) \in \mathscr{A} \\ 0 & \text{otherwise.} \end{cases} \tag{2.13}$$

Now Step 2 states that $q(y \mid x)$ equals $1/(Cg(x))$ for $y \in [0, Cg(x)]$ and is zero otherwise. Therefore, $q(x, y) = C^{-1}$ for every $(x, y) \in \mathscr{A}$.

Let (X^*, Y^*) be the first accepted point, that is, the first one that is in $\mathscr{B}$. Since the vector (X, Y) is uniformly distributed on $\mathscr{A}$, then clearly the vector (X^*, Y^*) is uniformly distributed on $\mathscr{B}$. Also, since the area of $\mathscr{B}$ equals unity, the joint pdf of (X^*, Y^*) on $\mathscr{B}$ equals unity as well. Thus, the marginal pdf of $Z = X^*$ is

$$\int_0^{f(x)} 1\, dy = f(x)\,.$$

□

The *efficiency* of Algorithm 2.3.5 is defined as

$$\mathbb{P}\left((X, Y) \text{ is accepted}\right) = \frac{\text{area } \mathscr{B}}{\text{area } \mathscr{A}} = \frac{1}{C}\,. \tag{2.14}$$

Often, a slightly modified version of Algorithm 2.3.5 is used. Namely, taking into account that $Y \sim \mathsf{U}(0, C\,g(X))$ in Step 2 is the same as setting $Y = U\,C\,g(X)$, where $U \sim \mathsf{U}(0,1)$, we can write $Y \leqslant f(X)$ in Step 3 as $U \leqslant f(X)/(C\,g(X))$. Thus, the modified version of Algorithm 2.3.5 can be rewritten as follows.

Algorithm 2.3.6 (Modified Acceptance-Rejection Algorithm)

1. *Generate X from $g(x)$.*
2. *Generate U from $\mathsf{U}(0,1)$ independently of X.*
3. *If $U \leqslant f(X)/(C\,g(X))$, return $Z = X$. Otherwise, return to Step 1.*

In other words, generate X from $g(x)$ and accept it with probability $f(X)/(C\,g(X))$; otherwise, reject X and try again.

■ **EXAMPLE 2.5 Example 2.2 (Continued)**

We shall show how to generate a random variable Z from the pdf

$$f(x) = \begin{cases} 2\,x, & 0 \leqslant x \leqslant 1 \\ 0 & \text{otherwise} \end{cases}$$

using the acceptance-rejection method. For simplicity, take $g(x) = 1$, $0 \leqslant x \leqslant 1$, and $C = 2$. That is, our proposal distribution is simply the uniform distribution on $[0,1]$. In this case, $f(x)/(C\,g(x)) = x$ and Algorithm 2.3.6 becomes:

1. *Generate X from $\mathsf{U}(0,1)$.*
2. *Generate U from $\mathsf{U}(0,1)$ independently of U.*
3. *If $U \leqslant X$, return $Z = X$. Otherwise, go to Step 1.*

Note that this example is merely illustrative, since the inverse-transform method handles it efficiently.

As a consequence of (2.14), the efficiency of the modified acceptance-rejection method is again determined by the acceptance probability $p = \mathbb{P}(U \leqslant f(X)/(Cg(X))) = \mathbb{P}(Y \leqslant f(X)) = 1/C$ for each trial (X, U). Since the trials are independent, the number of trials, N, before a successful pair (Z, U) occurs has the following geometric distribution:

$$\mathbb{P}(N = n) = p\,(1-p)^{n-1}, \quad n = 1, 2, \ldots, \tag{2.15}$$

with the expected number of trials equal to $1/p = C$.

For this method to be of practical interest, the following criteria must be used in selecting the proposal density $g(x)$:

1. It should be easy to generate a random variable from $g(x)$.
2. The efficiency, $1/C$, of the procedure should be large, that is, C should be close to 1 (which occurs when $g(x)$ is close to $f(x)$).

■ **EXAMPLE 2.6**

Generate a random variable Z from the semicircular density

$$f(x) = \frac{2}{\pi R^2}\sqrt{R^2 - x^2}, \quad -R \leqslant x \leqslant R .$$

Take the proposal distribution to be uniform over $[-R, R]$, that is, take $g(x) = 1/(2R)$, $-R \leqslant x \leqslant R$ and choose C as small as possible such that $Cg(x) \geqslant f(x)$; hence, $C = 4/\pi$. Then Algorithm 2.3.6 leads to the following generation algorithm:

1. *Generate two independent random variables, U_1 and U_2, from* $\mathsf{U}(0,1)$.
2. *Use U_2 to generate X from $g(x)$ via the inverse-transform method, namely, $X = (2U_2 - 1)R$, and calculate*

$$\frac{f(X)}{C\,g(X)} = \sqrt{1 - (2U_2 - 1)^2} .$$

3. *If $U_1 \leqslant f(X)/(C\,g(X))$, which is equivalent to $(2U_2 - 1)^2 \leqslant 1 - U_1^2$, return $Z = X = (2U_2 - 1)R$; otherwise, return to Step 1.*

The expected number of trials for this algorithm is $C = 4/\pi$, and the efficiency is $1/C = \pi/4 \approx 0.785$.

2.4 GENERATING FROM COMMONLY USED DISTRIBUTIONS

The next two subsections present algorithms for generating variables from commonly used continuous and discrete distributions. Of the numerous algorithms available (for example, [2]), we have tried to select those that are reasonably efficient and relatively simple to implement.

2.4.1 Generating Continuous Random Variables

2.4.1.1 Exponential Distribution We start by applying the inverse-transform method to the exponential distribution. If $X \sim \mathsf{Exp}(\lambda)$, then its cdf F is given by

$$F(x) = 1 - \mathrm{e}^{-\lambda x}, \qquad x \geqslant 0 . \tag{2.16}$$

Hence, solving $u = F(x)$ in terms of x gives

$$F^{-1}(u) = -\frac{1}{\lambda}\ln(1 - u) .$$

Noting that $U \sim \mathsf{U}(0,1)$ implies $1 - U \sim \mathsf{U}(0,1)$, we obtain the following algorithm.

Algorithm 2.4.1 (Generation of an Exponential Random Variable)

1. *Generate* $U \sim \mathsf{U}(0,1)$.
2. *Return* $X = -\frac{1}{\lambda}\ln U$ *as a random variable from* $\mathsf{Exp}(\lambda)$.

There are many alternative procedures for generating variables from the exponential distribution. The interested reader is referred to [2].

2.4.1.2 Normal (Gaussian) Distribution If $X \sim \mathsf{N}(\mu, \sigma^2)$, its pdf is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}, \qquad -\infty < x < \infty, \tag{2.17}$$

where μ is the mean (or expectation) and σ^2 the variance of the distribution.

Since inversion of the normal cdf is numerically inefficient, the inverse-transform method is not very suitable for generating normal random variables, and other procedures must be devised. We consider only generation from $\mathsf{N}(0,1)$ (standard normal variables), since any random $Z \sim \mathsf{N}(\mu, \sigma^2)$ can be represented as $Z = \mu + \sigma X$, where X is from $\mathsf{N}(0,1)$. One of the earliest methods for generating variables from $\mathsf{N}(0,1)$ is the following, developed by Box and Müller.

Let X and Y be two independent standard normal random variables, so (X, Y) is a random point in the plane. Let (R, Θ) be the corresponding polar coordinates. The joint pdf $f_{R,\Theta}$ of R and Θ is given by

$$f_{R,\Theta}(r,\theta) = \frac{1}{2\pi} \mathrm{e}^{-r^2/2} r \quad \text{for } r \geqslant 0 \text{ and } \theta \in [0, 2\pi).$$

This can be seen as follows. Specifying x and y in terms of r and θ gives

$$x = r\cos\theta \quad \text{and} \quad y = r\sin\theta. \tag{2.18}$$

The Jacobian of this coordinate transformation is

$$\det\begin{pmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{pmatrix} = \begin{vmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{vmatrix} = r.$$

The result now follows from the transformation rule (1.20), noting that the joint pdf of X and Y is $f_{X,Y}(x,y) = \frac{1}{2\pi}\mathrm{e}^{-(x^2+y^2)/2}$. It is not difficult to verify that R and Θ are independent, that $\Theta \sim \mathsf{U}[0, 2\pi)$, and that $\mathbb{P}(R > r) = \mathrm{e}^{-r^2/2}$. This means that R has the same distribution as $\sqrt{V}$, with $V \sim \mathsf{Exp}(1/2)$. Namely, $\mathbb{P}(\sqrt{V} > v) = \mathbb{P}(V > v^2) = \mathrm{e}^{-v^2/2}$, $v \geqslant 0$. Thus, both Θ and R are easy to generate and are transformed via (2.18) into independent standard normal random variables. This leads to the following algorithm.

Algorithm 2.4.2 (Generation of a Normal Random Variable: Box–Müller Approach)

1. *Generate two independent random variables, U_1 and U_2, from* $\mathsf{U}(0,1)$.
2. *Return two independent standard normal variables, X and Y, via*

$$\begin{aligned} X &= (-2\ln U_1)^{1/2}\cos(2\pi U_2), \\ Y &= (-2\ln U_1)^{1/2}\sin(2\pi U_2). \end{aligned} \tag{2.19}$$

An alternative generation method for $\mathsf{N}(0,1)$ is based on the acceptance-rejection method. First, note that in order to generate a random variable Y from $\mathsf{N}(0,1)$, one can first generate a positive random variable X from the pdf

$$f(x) = \sqrt{\frac{2}{\pi}}\,\mathrm{e}^{-x^2/2}, \qquad x \geqslant 0, \tag{2.20}$$

and then assign to X a random sign. The validity of this procedure follows from the symmetry of the standard normal distribution about zero.

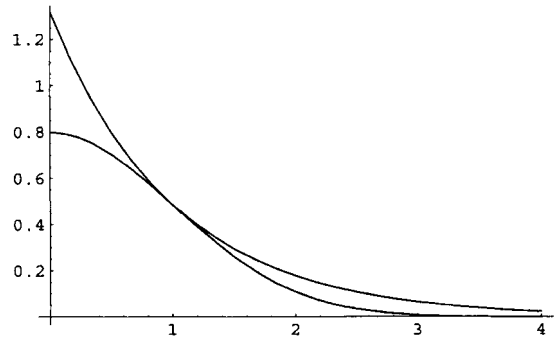


Figure 2.5 Bounding the positive normal density.

To generate a random variable X from (2.20), we bound $f(x)$ by $C\,g(x)$, where $g(x) = \mathrm{e}^{-x}$ is the pdf of the Exp(1) distribution. The smallest constant C such that $f(x) \leqslant Cg(x)$ is $\sqrt{2\mathrm{e}/\pi}$ (see Figure 2.5). The efficiency of this method is therefore $\sqrt{\pi/2\mathrm{e}} \approx 0.76$.

The acceptance condition, $U \leqslant f(X)/(C\mathrm{e}^{-X})$, can be written as

$$U \leqslant \exp[-(X-1)^2/2]\,, \tag{2.21}$$

which is equivalent to

$$-\ln U \geqslant \frac{(X-1)^2}{2}\,, \tag{2.22}$$

where X is from Exp(1). Since $-\ln U$ is also from Exp(1), the last inequality can be written as

$$V_1 \geqslant \frac{(V_2-1)^2}{2}\,, \tag{2.23}$$

where $V_1 = -\ln U$ and $V_2 = X$ are independent and both Exp(1) distributed.

2.4.1.3 Gamma Distribution If $X \sim \mathsf{Gamma}(\alpha, \lambda)$ then its pdf is of the form

$$f(x) = \frac{x^{\alpha-1}\lambda^{\alpha}\mathrm{e}^{-\lambda x}}{\Gamma(\alpha)}\,, \qquad x \geqslant 0\,. \tag{2.24}$$

The parameters $\alpha > 0$ and $\lambda > 0$ are called the *shape* and *scale* parameters, respectively. Since λ merely changes the scale, it suffices to consider only random variable generation of $\mathsf{Gamma}(\alpha, 1)$. In particular, if $X \sim \mathsf{Gamma}(\alpha, 1)$, then $X/\lambda \sim \mathsf{Gamma}(\alpha, \lambda)$ (see Exercise 2.16). Because the cdf for the gamma distributions does not generally exist in explicit form, the inverse-transform method cannot always be applied to generate random variables from this distribution. Alternative methods are thus called for. We discuss one such method for the case $\alpha \geqslant 1$. Let $f(x) = x^{\alpha-1}\mathrm{e}^{-x}/\Gamma(\alpha)$ and $\psi(x) = d(1+cx)^3$, $x > -1/c$ and zero otherwise, where c and d are positive constants. Note that $\psi(x)$ is a strictly increasing function. Let Y have density $k(y) = f(\psi(y))\,\psi'(y)\,c_1$, where c_1 is a normalization constant. Then $X = \psi(Y)$ has density f. Namely, by the transformation rule (1.16)

$$f_X(x) = \frac{k(\psi^{-1}(x))}{\psi'(\psi^{-1}(x))} = f(\psi(\psi^{-1}(x)))\,\frac{\psi'(\psi^{-1}(x))}{\psi'(\psi^{-1}(x))} = f(x)\,.$$

We draw Y via the acceptance-rejection method, using the standard normal distribution as our proposal distribution. We choose c and d such that $k(y) \leqslant C\varphi(y)$, with $C > 1$ close to 1, where φ is the pdf of the $\mathsf{N}(0,1)$ distribution. To find such c and d, first write $k(y) = c_2\, \mathrm{e}^{h(y)}$, where some algebra will show that

$$h(y) = (1 - 3\alpha)\ln(1 + cy) - d(1 + cy)^3 + d\,.$$

(Note that $h(0) = 0$.) Next, a Taylor series expansion of $h(y)$ around 0 yields

$$h(y) = c(-1 - 3d + 3\alpha)\, y - \frac{1}{2}\, c^2 (-1 + 6d + 3\alpha)\, y^2 + O(y^3)\,.$$

This suggests taking c and d such that the coefficients of y and y^2 in the expansion above are 0 and $-1/2$, respectively, as in the exponent of the standard normal density. This gives $d = \alpha - 1/3$ and $c = \frac{1}{3\sqrt{d}}$. It is not difficult to check that indeed

$$h(y) \leqslant -\frac{1}{2}\, y^2 \quad \text{for all} \quad y > -\frac{1}{c}\,,$$

so that $\mathrm{e}^{h(y)} \leqslant \mathrm{e}^{-\frac{1}{2}y^2}$, which means that $k(y)$ is dominated by $c_2\sqrt{2\pi}\,\varphi(y)$ for all y. Hence, the acceptance-rejection method for drawing from $Y \sim k$ is as follows: Draw $Z \sim \mathsf{N}(0,1)$ and $U \sim \mathsf{U}(0,1)$ independently. If

$$U < \frac{c_2\, \mathrm{e}^{h(Z)}}{c_2\sqrt{2\pi}\,\varphi(Z)}\,,$$

or equivalently, if

$$\ln U < h(Z) + \frac{1}{2} Z^2\,,$$

then return $Y = Z$; otherwise, repeat (we set $h(Z) = -\infty$ if $Z \leqslant -1/c$). The efficiency of this method, $\int_{-1/c}^{\infty} \mathrm{e}^{h(y)} dy / \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2}y^2}\, dy$, is greater than 0.95 for all values of $\alpha \geqslant 1$. Finally, we complete the generation of X by taking $X = \psi(Y)$. Summarizing, we have the following algorithm [8].

Algorithm 2.4.3 (Sampling from the Gamma$(\alpha, 1)$ Distribution ($\alpha \geqslant 1$))

1. *Set* $d = \alpha - 1/3$ *and* $c = 1/\sqrt{9\,d}$.
2. *Generate* $Z \sim \mathsf{N}(0,1)$.
3. *Generate* $U \sim \mathsf{U}(0,1)$.
4. *If* $Z > -1/c$ *and* $\ln U < h(Z) + \frac{1}{2}Z^2$, *return* $X = d(1 + cZ)^3$; *otherwise, go back to Step 2.*

For the case where $\alpha < 1$, one can use the fact that if $X \sim \mathsf{Gamma}(1 + \alpha, 1)$ and $U \sim \mathsf{U}(0,1)$ are independent, then $XU^{1/\alpha} \sim \mathsf{Gamma}(\alpha, 1)$; see Problem 2.17.

A gamma distribution with an integer shape parameter, say $\alpha = m$, is also called an *Erlang distribution*, denoted $\mathsf{Erl}(m, \lambda)$. In this case, X can be represented as the sum of iid exponential random variables Y_i. That is, $X = \sum_{i=1}^{m} Y_i$, where the $\{Y_i\}$ are iid

exponential variables, each with mean $1/\lambda$; see Example 1.9. Using Algorithm 2.4.1, we can write $Y_i = -\frac{1}{\lambda} \ln U_i$, whence

$$X = -\frac{1}{\lambda} \sum_{i=1}^{m} \ln U_i = -\frac{1}{\lambda} \ln \prod_{i=1}^{m} U_i \,. \tag{2.25}$$

Equation (2.25) suggests the following generation algorithm.

Algorithm 2.4.4 (Generation of an Erlang Random Variable)

1. *Generate iid random variables* $U_1, \ldots, U_m$ *from* $\mathsf{U}(0,1)$.
2. *Return* $X = -\frac{1}{\lambda} \ln \prod_{i=1}^{m} U_i$.

2.4.1.4 Beta Distribution If $X \sim \mathsf{Beta}(\alpha, \beta)$, then its pdf is of the form

$$f(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad 0 \leqslant x \leqslant 1 \,. \tag{2.26}$$

Both parameters α and β are assumed to be greater than 0. Note that $\mathsf{Beta}(1,1)$ is simply the $\mathsf{U}(0,1)$ distribution.

To sample from the beta distribution, consider first the case where either α or β equals 1. In that case, one can simply use the inverse-transform method. For example, for $\beta = 1$, the $\mathsf{Beta}(\alpha, 1)$ pdf is

$$f(x) = \alpha x^{\alpha-1}, \quad 0 \leqslant x \leqslant 1 \,,$$

and the corresponding cdf becomes

$$F(x) = x^{\alpha}, \quad 0 \leqslant x \leqslant 1 \,.$$

Thus, a random variable X can be generated from this distribution by drawing $U \sim \mathsf{U}(0,1)$ and returning $X = U^{1/\alpha}$.

A general procedure for generating a $\mathsf{Beta}(\alpha, \beta)$ random variable is based on the fact that if $Y_1 \sim \mathsf{Gamma}(\alpha, 1)$, $Y_2 \sim \mathsf{Gamma}(\beta, 1)$, and Y_1 and Y_2 are independent, then

$$X = \frac{Y_1}{Y_1 + Y_2}$$

is distributed $\mathsf{Beta}(\alpha, \beta)$. The reader is encouraged to prove this assertion (see Problem 2.18). The corresponding algorithm is as follows.

Algorithm 2.4.5 (Generation of a Beta Random Variable)

1. *Generate independently* $Y_1 \sim \mathsf{Gamma}(\alpha, 1)$ *and* $Y_2 \sim \mathsf{Gamma}(\beta, 1)$.
2. *Return* $X = Y_1/(Y_1 + Y_2)$ *as a random variable from* $\mathsf{Beta}(\alpha, \beta)$.

For integer $\alpha = m$ and $\beta = n$, another method may be used, based on the theory of order statistics. Let $U_1, \ldots, U_{m+n-1}$ be independent random variables from $\mathsf{U}(0,1)$. Then the m-th order statistic, $U_{(m)}$, has a $\mathsf{Beta}(m, n)$ distribution. This gives the following algorithm.

Algorithm 2.4.6 (Generation of a Beta Random Variable with Integer Parameters $\alpha = m$ and $\beta = n$)

1. *Generate* $m + n - 1$ *iid random variables* $U_1, \ldots, U_{m+n-1}$ *from* $\mathsf{U}(0,1)$.
2. *Return the* m*-th order statistic,* $U_{(m)}$*, as a random variable from* $\mathsf{Beta}(m, n)$.

It can be shown that the total number of comparisons needed to find $U_{(m)}$ is $(m/2)(m + 2n - 1)$, so that this procedure loses efficiency for large m and n.

2.4.2 Generating Discrete Random Variables

2.4.2.1 Bernoulli Distribution If $X \sim \mathsf{Ber}(p)$, its pdf is of the form

$$f(x) = p^x(1-p)^{1-x}, \qquad x = 0, 1\,, \tag{2.27}$$

where p is the success probability. Applying the inverse-transform method, one readily obtains the following generation algorithm.

Algorithm 2.4.7 (Generation of a Bernoulli Random Variable)

1. *Generate* $U \sim \mathsf{U}(0,1)$.
2. *If* $U \leqslant p$, *return* $X = 1$; *otherwise, return* $X = 0$.

In Figure 2.6, three typical outcomes (realizations) are given for 100 independent Bernoulli random variables, each with success parameter $p = 0.5$.

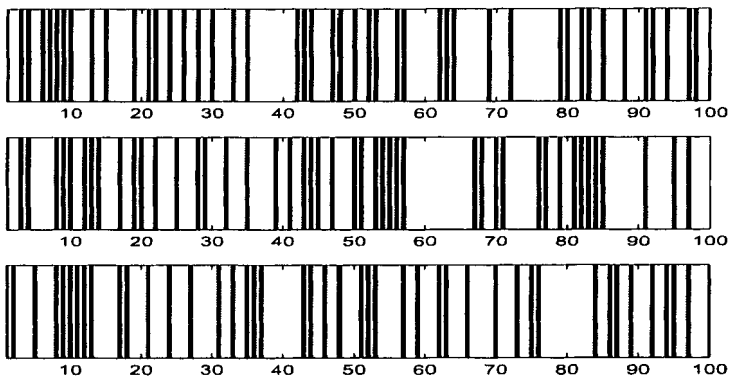


Figure 2.6 Results of three experiments with 100 independent Bernoulli trials, each with $p = 0.5$. The dark bars indicate where a success appears.

2.4.2.2 Binomial Distribution If $X \sim \mathsf{Bin}(n, p)$ then its pdf is of the form

$$f(x) = \binom{n}{x} p^x(1-p)^{n-x}, \quad x = 0, 1, \ldots, n\,. \tag{2.28}$$

Recall that a binomial random variable X can be viewed as the total number of successes in n independent Bernoulli experiments, each with success probability p; see Example 1.1. Denoting the result of the i-th trial by $X_i = 1$ (success) or $X_i = 0$ (failure), we can write $X = X_1 + \cdots + X_n$, with the $\{X_i\}$ being iid $\mathsf{Ber}(p)$ random variables. The simplest generation algorithm can thus be written as follows.

Algorithm 2.4.8 (Generation of a Binomial Random Variable)

1. *Generate iid random variables* $X_1, \ldots, X_n$ *from* $\mathsf{Ber}(p)$.
2. *Return* $X = \sum_{i=1}^{n} X_i$ *as a random variable from* $\mathsf{Bin}(n, p)$.

Since the execution time of Algorithm 2.4.8 is proportional to n, one is motivated to use alternative methods for large n. For example, one may consider the normal distribution as an approximation to the binomial. In particular, by the central limit theorem, as n increases, the distribution of X is close to that of $Y \sim \mathsf{N}(np, np(1-p))$; see (1.26). In fact, the cdf of $\mathsf{N}(np - 1/2, np(1-p))$ approximates the cdf of X even better. This is called the *continuity correction*.

Thus, to obtain a binomial random variable, we generate Y from $\mathsf{N}(np - 1/2, np(1-p))$ and truncate to the nearest nonnegative integer. Equivalently, we generate $Z \sim \mathsf{N}(0,1)$ and set

$$\max\left\{0, \left\lfloor np - \frac{1}{2} + Z\sqrt{np(1-p)} \right\rfloor\right\} \tag{2.29}$$

as an approximate sample from the $\mathsf{Bin}(n,p)$ distribution. Here $\lfloor \alpha \rfloor$ denotes the integer part of α. One should consider using the normal approximation for $np > 10$ with $p \geqslant \frac{1}{2}$, and for $n(1-p) > 10$ with $p < \frac{1}{2}$.

It is worthwhile to note that if $Y \sim \mathsf{Bin}(n,p)$, then $n - Y \sim \mathsf{Bin}(n, 1-p)$. Hence, to enhance efficiency, one may elect to generate X from $\mathsf{Bin}(n,p)$ according to

$$X = \begin{cases} Y_1 \sim \mathsf{Bin}(n,p) & \text{if } p \leqslant \frac{1}{2} \\ Y_2 \sim \mathsf{Bin}(n, 1-p) & \text{if } p > \frac{1}{2} \,. \end{cases}$$

2.4.2.3 *Geometric Distribution* If $X \sim \mathsf{G}(p)$, then its pdf is of the form

$$f(x) = p\,(1-p)^{x-1}, \qquad x = 1, 2 \ldots . \tag{2.30}$$

The random variable X can be interpreted as the number of trials required until the first success occurs in a series of independent Bernoulli trials with success parameter p. Note that $\mathbb{P}(X > m) = (1-p)^m$.

We now present an algorithm based on the relationship between the exponential and geometric distributions. Let $Y \sim \mathsf{Exp}(\lambda)$, with λ such that $1 - p = \mathrm{e}^{-\lambda}$. Then $X = \lfloor Y \rfloor + 1$ has a $\mathsf{G}(p)$ distribution. This is because

$$\mathbb{P}(X > x) = \mathbb{P}(\lfloor Y \rfloor > x - 1) = \mathbb{P}(Y \geqslant x) = \mathrm{e}^{-\lambda x} = (1-p)^x \,.$$

Hence, to generate a random variable from $\mathsf{G}(p)$, we first generate a random variable from the exponential distribution with $\lambda = -\ln(1-p)$, truncate the obtained value to the nearest integer, and add 1.

Algorithm 2.4.9 (Generation of a Geometric Random Variable)

1. *Generate* $Y \sim \mathsf{Exp}(-\ln(1-p))$.
2. *Return* $X = 1 + \lfloor Y \rfloor$ *as a random variable from* $\mathsf{G}(p)$.

2.4.2.4 *Poisson Distribution* If $X \sim \mathsf{Poi}(\lambda)$, its pdf is of the form

$$f(n) = \frac{\mathrm{e}^{-\lambda}\,\lambda^n}{n!}, \qquad n = 0, 1, \ldots , \tag{2.31}$$

where λ is the *rate* parameter. There is an intimate relationship between Poisson and exponential random variables, highlighted by the properties of the Poisson process; see Section 1.11. In particular, a Poisson random variable X can be interpreted as the maximal

number of iid exponential variables (with parameter λ) whose sum does not exceed 1. That is,

$$X = \max\left\{ n : \sum_{j=1}^{n} Y_j \leqslant 1 \right\}, \tag{2.32}$$

where the $\{Y_i\}$ are independent and $\mathsf{Exp}(\lambda)$ distributed. Since $Y_j = -\frac{1}{\lambda}\ln U_j$, with $U_j \sim \mathsf{U}(0,1)$, we can rewrite (2.32) as

$$\begin{aligned} X &= \max\left\{ n : \sum_{j=1}^{n} -\ln U_j \leqslant \lambda \right\} \\ &= \max\left\{ n : \ln\left(\prod_{j=1}^{n} U_j \right) \geqslant -\lambda \right\} \\ &= \max\left\{ n : \prod_{j=1}^{n} U_j \geqslant \mathrm{e}^{-\lambda} \right\}. \end{aligned} \tag{2.33}$$

This leads to the following algorithm.

Algorithm 2.4.10 (Generating a Poisson Random Variable)

1. *Set* $n = 1$ *and* $a = 1$.
2. *Generate* $U_n \sim \mathsf{U}(0,1)$ *and set* $a = a\,U_n$.
3. *If* $a \geqslant e^{-\lambda}$, *then set* $n = n + 1$ *and go to Step 2.*
4. *Otherwise, return* $X = n - 1$ *as a random variable from* $\mathsf{Poi}(\lambda)$.

It is readily seen that for large λ, this algorithm becomes slow ($\mathrm{e}^{-\lambda}$ is small for large λ, and more random numbers, U_j, are required to satisfy $\prod_{j=1}^{n} U_j < \mathrm{e}^{-\lambda}$). Alternative approaches use the inverse-transform method with an efficient search (see [2]) or the alias method.

2.5 RANDOM VECTOR GENERATION

Suppose that we wish to generate a random vector $\mathbf{X} = (X_1, \ldots, X_n)$ from a given n-dimensional distribution with pdf $f(\mathbf{x})$ and cdf $F(\mathbf{x})$. When the components $X_1, \ldots, X_n$ are *independent*, the situation is easy: we simply apply the inverse-transform method or another generation method of our choice to each component individually.

■ **EXAMPLE 2.7**

We wish to generate uniform random vectors $\mathbf{X} = (X_1, \ldots, X_n)$ from the n-dimensional rectangle $D = \{(x_1, \ldots, x_n) : a_i \leqslant x_i \leqslant b_i,\ i = 1, \ldots, n\}$. It is clear that this implies that the components of $\mathbf{X}$ are independent and uniformly distributed: $X_i \sim U[a_i, b_i]$, $i = 1, \ldots, n$. Applying the inverse-transform method to X_i, we can write $X_i = a_i + (b_i - a_i)\,U_i$, $i = 1, \ldots, n$, where $U_1, \ldots, U_n$ are iid from $\mathsf{U}(0,1)$.

For *dependent* random variables $X_1, \ldots, X_n$, we can represent the joint pdf $f(\mathbf{x})$, using the product rule (1.4), as

$$f(x_1, \ldots, x_n) = f_1(x_1)\, f_2(x_2 \mid x_1) \cdots f_n(x_n \mid x_1, \ldots, x_{n-1}) , \tag{2.34}$$

where $f_1(x_1)$ is the marginal pdf of X_1 and $f_k(x_k \mid x_1, \ldots, x_{k-1})$ is the conditional pdf of X_k given $X_1 = x_1, X_2 = x_2, \ldots, X_{k-1} = x_{k-1}$. Thus, one way to generate $\mathbf{X}$ is to first generate X_1, then, given $X_1 = x_1$, generate X_2 from $f_2(x_2 \mid x_1)$, and so on, until we generate X_n from $f_n(x_n \mid x_1, \ldots, x_{n-1})$.

The applicability of this approach depends, of course, on knowledge of the conditional distributions. In certain models, for example Markov models, this knowledge is easily obtainable.

2.5.1 Vector Acceptance-Rejection Method

The acceptance-rejection Algorithm 2.3.6 is directly applicable to the multidimensional case. We need only bear in mind that the random variable X (see Step 2 of Algorithm 2.3.6) becomes an n-dimensional random vector $\mathbf{X}$. Consequently, we need a convenient way of generating $\mathbf{X}$ from the multidimensional proposal pdf $g(\mathbf{x})$, for example, by using the vector inverse-transform method. The following example demonstrates the vector version of the acceptance-rejection method.

■ **EXAMPLE 2.8**

We want to generate a random vector $\mathbf{Z}$ that is uniformly distributed over an irregular n-dimensional region G (see Figure 2.7). The algorithm is straightforward:

1. *Generate a random vector, $\mathbf{X}$, uniformly distributed in W, where W is a regular region (multidimensional hypercube, hyperrectangle, hypersphere, hyperellipsoid, etc.).*
2. *If $\mathbf{X} \in G$, accept $\mathbf{Z} = \mathbf{X}$ as the random vector uniformly distributed over G; otherwise, return to Step 1.*

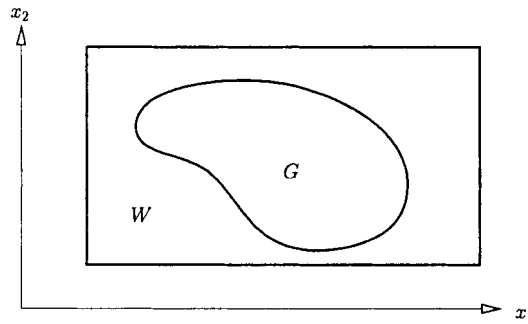


Figure 2.7 The vector acceptance-rejection method.

As a special case, let G be the n-dimensional unit ball, that is, $G = \{\mathbf{x} : \sum_i x_i^2 \leqslant 1\}$, and let W be the n-dimensional hypercube $\{-1 \leqslant x_i \leqslant 1\}_{i=1}^n$. To generate a random

vector that is uniformly distributed over the interior of the n-dimensional unit ball, we generate a random vector $\mathbf{X}$ that is uniformly distributed over W and then accept or reject it, depending on whether it falls inside or outside the n-dimensional ball. The corresponding algorithm is as follows.

Algorithm 2.5.1

1. *Generate* $U_1, \ldots, U_n$ *as iid random variables from* $\mathsf{U}(0,1)$.
2. *Set* $X_1 = 1 - 2U_1, \ldots, X_n = 1 - 2U_n$, *and* $R = \sum_{i=1}^n X_i^2$.
3. *If* $R \leqslant 1$, *accept* $\mathbf{X} = (X_1, \ldots, X_n)$ *as the desired vector; otherwise, go to Step 1.*

Remark 2.5.1 To generate a random vector that is uniformly distributed over the *surface* of an n-dimensional unit ball — in other words, uniformly over the unit sphere $\{\mathbf{x} : \sum_i x_i^2 = 1\}$ — we need only rewrite Step 3 in Algorithm 2.5.1 as follows:

3′. *If* $R \leqslant 1$, *accept* $\mathbf{Z} = (Z_1, \ldots, Z_n)$ *with* $Z_i = X_i/\sqrt{R}$, $i = 1, \ldots, n$, *as the desired vector.*

The efficiency of the vector acceptance-rejection method is equal to the ratio

$$\frac{1}{C} = \frac{\text{volume of the hyperball}}{\text{volume of the hypercube}} = \frac{1}{n\,2^{n-1}} \frac{\pi^{n/2}}{\Gamma(n/2)},$$

where the volumes of the ball and cube are $\frac{\pi^{n/2}}{(n/2)\Gamma(n/2)}$ and 2^n, respectively. Note that for even n ($n = 2m$) we have

$$\frac{1}{C} = \frac{\pi^m}{m!\,2^{2m}} = \frac{1}{m!}\left(\frac{\pi}{2}\right)^m 2^{-m} \to 0 \text{ as } m \to \infty.$$

In other words, the acceptance-rejection method grows inefficient in n, and is asymptotically useless.

2.5.2 Generating Variables from a Multinormal Distribution

The key to generating a multivariate normal (or simply multinormal) random vector $\mathbf{Z} \sim \mathsf{N}(\boldsymbol{\mu}, \Sigma)$ is to write it as $\mathbf{Z} = \boldsymbol{\mu} + B\mathbf{X}$, where B a matrix such that $BB^T = \Sigma$, and $\mathbf{X}$ is a vector of iid $\mathsf{N}(0,1)$ random variables; see Section 1.9. Note that $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ is the mean vector and Σ is the $(n \times n)$ covariance matrix of $\mathbf{Z}$. For any covariance matrix Σ, such a matrix B can always be found efficiently using the Cholesky square root method; see Section A.1 of the Appendix.

The following algorithm describes the generation of a $\mathsf{N}(\boldsymbol{\mu}, \Sigma)$ distributed random vector $\mathbf{Z}$.

Algorithm 2.5.2 (Generation of Multinormal Vectors)

1. *Generate* $X_1, \ldots, X_n$ *as iid variables from* $\mathsf{N}(0,1)$.
2. *Derive the Cholesky decomposition* $\Sigma = BB^T$.
3. *Return* $\mathbf{Z} = \boldsymbol{\mu} + B\mathbf{X}$.

2.5.3 Generating Uniform Random Vectors Over a Simplex

Consider the n-dimensional simplex,

$$\mathscr{Y} = \left\{ \mathbf{y} : y_i \geqslant 0, \;\; i = 1, \ldots, n, \; \sum_{i=1}^{n} y_i \leqslant 1 \right\} . \tag{2.35}$$

$\mathscr{Y}$ is a simplex on the points $\mathbf{0}, \mathbf{e}_1, \ldots, \mathbf{e}_n$, where $\mathbf{0}$ is the zero vector and $\mathbf{e}_i$ is the i-th unit vector in $\mathbb{R}^n$, $i = 1, \ldots, n$. Let $\mathscr{X}$ be a second n-dimensional simplex:

$$\mathscr{X} = \{ \mathbf{x} : x_i \geqslant 0, \;\; i = 1, \ldots, n, \; x_1 \leqslant x_2 \leqslant \cdots \leqslant x_n \leqslant 1 \} .$$

$\mathscr{X}$ is a simplex on the points $\mathbf{0}, \mathbf{e}_n, \mathbf{e}_n + \mathbf{e}_{n-1}, \ldots, \mathbf{1}$, where $\mathbf{1}$ is the sum of all unit vectors (a vector of 1s). Figure 2.8 illustrates the two-dimensional case.

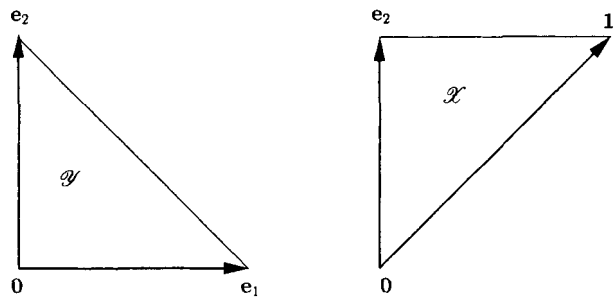


Figure 2.8 Simplexes $\mathscr{Y}$ and $\mathscr{X}$.

Simplex $\mathscr{Y}$ can be obtained from simplex $\mathscr{X}$ by the linear transformation $\mathbf{y} = A\mathbf{x}$ with

$$A = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 1 \end{pmatrix} .$$

Now, drawing a vector $\mathbf{X} = (X_1, \ldots, X_n)$ according to the uniform distribution on $\mathscr{X}$ is easy: simply take X_i to be the i-th order statistic of iid random variables $U_1, \ldots, U_n$ from $\mathrm{U}(0, 1)$. Since a linear transformation preserves uniformity, applying matrix A to $\mathbf{X}$ yields a vector $\mathbf{Y}$ that is uniformly distributed on $\mathscr{Y}$.

Algorithm 2.5.3 (Generating a Vector Over a Simplex)

1. *Generate n independent random variables $U_1, \ldots, U_n$ from* $\mathrm{U}(0, 1)$.
2. *Sort $U_1, \ldots, U_n$ into the order statistics $U_{(1)}, \ldots, U_{(n)}$.*
3. *Define*

$$\begin{aligned} Y_1 &= U_{(1)} , \\ Y_2 &= U_{(2)} - U_{(1)} , \\ &\vdots \\ Y_n &= U_{(n)} - U_{(n-1)} \end{aligned} \tag{2.36}$$

and return the vector $\mathbf{Y} = (Y_1, \ldots, Y_n)$.

If we define $Y_{n+1} = 1 - \sum_{i=1}^{n} Y_i = 1 - U_{(n)}$, then the resulting $(n+1)$-dimensional vector $(Y_1, \ldots, Y_{n+1})$ will be uniformly distributed over the set

$$\mathscr{S} = \left\{ \mathbf{y} : y_i \geqslant 0, \ \ i = 1, \ldots, n+1, \ \sum_{i=1}^{n+1} y_i = 1 \right\},$$

that is, over the dominant face of the simplex defined by the points $\mathbf{0}, \mathbf{e}_1, \ldots, \mathbf{e}_{n+1}$.

Finally, in order to generate random vectors uniformly distributed over an n-dimensional simplex defined by arbitrary vertices, say $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_n$, we simply generate $\mathbf{Y}$ uniformly on $\mathscr{Y}$ and apply the linear transformation

$$\mathbf{Z} = C\mathbf{Y} + \mathbf{z}_0,$$

where C is the matrix whose columns are $\mathbf{z}_1 - \mathbf{z}_0, \ldots, \mathbf{z}_n - \mathbf{z}_0$.

2.5.4 Generating Random Vectors Uniformly Distributed Over a Unit Hyperball and Hypersphere

Algorithm 2.5.1 and Remark 2.5.1 explain how, using the multidimensional acceptance-rejection method, one can generate random vectors that are uniformly distributed over an n-dimensional unit hyperball (or simply n-ball). By simply dividing each vector by its length, one obtains random vectors that are uniformly distributed over the *surface* of the n-ball, that is, the n-sphere. The main advantage of the acceptance-rejection method is its simplicity. Its main disadvantage is that the number of trials needed to generate points inside the n-ball increases explosively with n. For this reason, it can be recommended only for low dimensions ($n \leqslant 5$). An alternative algorithm is based on the following result.

Theorem 2.5.1 *Let $X_1, \ldots, X_n$ be iid random variables from $\mathsf{N}(0,1)$, and let $||\mathbf{X}|| = (\sum_{i=1}^{n} X_i^2)^{\frac{1}{2}}$. Then the vector*

$$\mathbf{Y} = \left(\frac{X_1}{||\mathbf{X}||}, \ldots, \frac{X_n}{||\mathbf{X}||} \right) \tag{2.37}$$

is distributed uniformly over the n-sphere $\{\mathbf{y} : ||\mathbf{y}|| = 1\}$.

Proof: Note that $\mathbf{Y}$ is simply the projection of $\mathbf{X} = (X_1, \ldots, X_n)$ onto the n-sphere. The fact that $\mathbf{Y}$ is uniformly distributed follows immediately from the fact that the pdf of $\mathbf{X}$ is spherically symmetrical: $f_{\mathbf{X}}(\mathbf{x}) = c\mathrm{e}^{-||\mathbf{x}||^2/2}$. □

To obtain uniform random variables within the n-ball, we simply multiply the vector $\mathbf{Y}$ by $U^{1/n}$, where $U \sim \mathsf{U}(0,1)$. To see this, note that for a random vector $\mathbf{Z} = (Z_1, \ldots, Z_n)$ that is uniformly distributed over the n-ball, the radius $R = ||\mathbf{Z}||$ satisfies $\mathbb{P}(R \leqslant r) = r^n$. Hence, by the inverse-transform method, we can write $R = U^{1/n}$. This motivates the following alternative.

Algorithm 2.5.4 (Generating Uniform Random Vectors Over the n-Ball)

1. *Generate a random vector $\mathbf{X} = (X_1, \ldots, X_n)$ with iid $\mathsf{N}(0,1)$ components.*
2. *Generate $R = U^{1/n}$, with $U \sim \mathsf{U}(0,1)$.*
2. *Return $\mathbf{Z} = R\mathbf{X}/||\mathbf{X}||$.*

2.5.5 Generating Random Vectors Uniformly Distributed Over a Hyperellipsoid

The equation for a hyperellipsoid, centered at the origin, can be written as

$$\mathbf{x}^T \Sigma \mathbf{x} = r^2 , \tag{2.38}$$

where Σ is a positive definite and symmetric $(n \times n)$ matrix ($\mathbf{x}$ is interpreted as a column vector). The special case where $\Sigma = I$ (identity matrix) corresponds to a hypersphere of radius r. Since Σ is positive definite and symmetric, there exists a unique lower triangular matrix B such that $\Sigma = BB^T$; see (1.25). We may thus view the set $\mathscr{X} = \{\mathbf{x} : \mathbf{x}^T \Sigma \mathbf{x} \leqslant r^2\}$ as a linear transformation $\mathbf{y} = B^T \mathbf{x}$ of the n-dimensional ball $\mathscr{Y} = \{\mathbf{y} : \mathbf{y}^T \mathbf{y} \leqslant r^2\}$. Since linear transformations preserve uniformity, if the vector $\mathbf{Y}$ is uniformly distributed over the interior of an n-dimensional sphere of radius r, then the vector $\mathbf{X} = (B^T)^{-1}\mathbf{Y}$ is uniformly distributed over the interior of a hyperellipsoid (see (2.38)). The corresponding generation algorithm is given below.

Algorithm 2.5.5 (Generating Random Vectors Over the Interior of a Hyperellipsoid)

1. *Generate* $\mathbf{Y} = (Y_1, \ldots, Y_n)$, *uniformly distributed over the n-sphere of radius* r.
2. *Calculate the matrix* B, *satisfying* $\Sigma = BB^T$.
3. *Return* $\mathbf{X} = (B^T)^{-1}\mathbf{Y}$ *as the required uniform random vector.*

2.6 GENERATING POISSON PROCESSES

This section treats the generation of Poisson processes. Recall from Section 1.11 that there are two different (but equivalent) characterizations of a Poisson process $\{N_t, t \geqslant 0\}$. In the first (see Definition 1.11.1), the process is interpreted as a counting measure, where N_t counts the number of arrivals in $[0, t]$. The second characterization is that the interarrival times $\{A_i\}$ of $\{N_t, t \geqslant 0\}$ form a *renewal process*, that is, a sequence of iid random variables. In this case the interarrival times have an $\mathsf{Exp}(\lambda)$ distribution, and we can write $A_i = -\frac{1}{\lambda} \ln U_i$, where the $\{U_i\}$ are iid $\mathsf{U}(0, 1)$ distributed. Using the second characterization, we can generate the arrival times $T_i = A_1 + \cdots + A_i$ during the interval $[0, T]$ as follows.

Algorithm 2.6.1 (Generating a Homogeneous Poisson Process)

1. *Set* $T_0 = 0$ *and* $n = 1$.
2. *Generate an independent random variable* $U_n \sim \mathsf{U}(0, 1)$.
3. *Set* $T_n = T_{n-1} - \frac{1}{\lambda} \ln U_n$ *and declare an arrival.*
4. *If* $T_n > T$, *stop; otherwise, set* $n = n + 1$ *and go to Step 2.*

The first characterization of a Poisson process, that is, as a random counting measure, provides an alternative way of generating such processes, which works also in the multidimensional case. In particular (see the end of Section 1.11), the following procedure can be used to generate a homogeneous Poisson process with rate λ on any set A with "volume" $|A|$.

Algorithm 2.6.2 (Generating an n-Dimensional Poisson Process)

1. *Generate a Poisson random variable* $N \sim \mathsf{Poi}(\lambda\,|A|)$.
2. *Given* $N = n$, *draw* n *points independently and uniformly in* A. *Return these as the points of the Poisson process.*

A *nonhomogeneous Poisson process* is a counting process $N = \{N_t, t \geqslant 0\}$ for which the number of points in nonoverlapping intervals are independent — similar to the ordinary Poisson process — but the rate at which points arrive is *time dependent.* If $\lambda(t)$ denotes the rate at time t, the number of points in any interval (b, c) has a Poisson distribution with mean $\int_b^c \lambda(t)\,dt$.

Figure 2.9 illustrates a way to construct such processes. We first generate a two-dimensional homogeneous Poisson process on the strip $\{(t, x), t \geqslant 0, 0 \leqslant x \leqslant \lambda\}$, with constant rate $\lambda = \max \lambda(t)$, and then simply project all points below the graph of $\lambda(t)$ onto the t-axis.

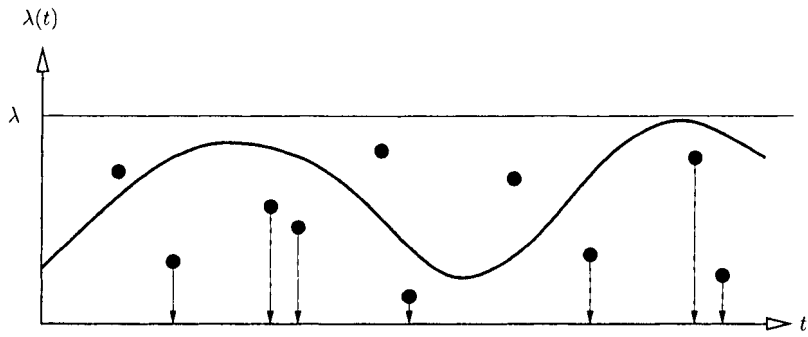


Figure 2.9 Constructing a nonhomogeneous Poisson process.

Note that the points of the two-dimensional Poisson process can be viewed as having a time and space dimension. The arrival epochs form a one-dimensional Poisson process with rate λ, and the positions are uniform on the interval $[0, \lambda]$. This suggests the following alternative procedure for generating nonhomogeneous Poisson processes: each arrival epoch of the one-dimensional homogeneous Poisson process is rejected (thinned) with probability $1 - \frac{\lambda(T_n)}{\lambda}$, where T_n is the arrival time of the n-th event. The surviving epochs define the desired nonhomogeneous Poisson process.

Algorithm 2.6.3 (Generating a Nonhomogeneous Poisson Process)

1. *Set* $t = 0$, $n = 0$ *and* $i = 0$.
2. *Increase* i *by 1.*
3. *Generate an independent random variable* $U_i \sim \mathsf{U}(0, 1)$.
4. *Set* $t = t - \frac{1}{\lambda} \ln U_i$.
5. *If* $t > T$, *stop; otherwise, continue.*
6. *Generate an independent random variable* $V_i \sim \mathsf{U}(0, 1)$.
7. *If* $V_i \leqslant \frac{\lambda(t)}{\lambda}$, *increase* n *by 1 and set* $T_n = t$. *Go to Step 2.*

2.7 GENERATING MARKOV CHAINS AND MARKOV JUMP PROCESSES

We now discuss how to simulate a Markov chain $X_0, X_1, X_2, \ldots, X_n$. To generate a Markov chain with initial distribution $\pi^{(0)}$ and transition matrix P, we can use the procedure outlined in Section 2.5 for dependent random variables. That is, first generate X_0 from $\pi^{(0)}$. Then, given $X_0 = x_0$, generate X_1 from the conditional distribution of X_1 given $X_0 = x_0$; in other words, generate X_1 from the x_0-th row of P. Suppose $X_1 = x_1$. Then, generate X_2 from the x_1-st row of P, and so on. The algorithm for a general discrete-state Markov chain with a one-step transition matrix P and an initial distribution vector $\pi^{(0)}$ is as follows:

Algorithm 2.7.1 (Generating a Markov Chain)

1. *Draw X_0 from the initial distribution $\pi^{(0)}$. Set $t = 0$.*
2. *Draw X_{t+1} from the distribution corresponding to the X_t-th row of P.*
3. *Set $t = t + 1$ and go to Step 2.*

■ **EXAMPLE 2.9 Random Walk on the Integers**

Consider the random walk on the integers in Example 1.10. Let $X_0 = 0$ (that is, we start at 0). Suppose the chain is at some discrete time $t = 0, 1, 2 \ldots$ in state i. Then, in Step 2 of Algorithm 2.7.1, we simply need to draw from a two-point distribution with mass p and q at $i + 1$ and $i - 1$, respectively. In other words, we draw $I_t \sim \text{Ber}(p)$ and set $X_{t+1} = X_t + 2I_t - 1$. Figure 2.10 gives a typical sample path for the case where $p = q = 1/2$.

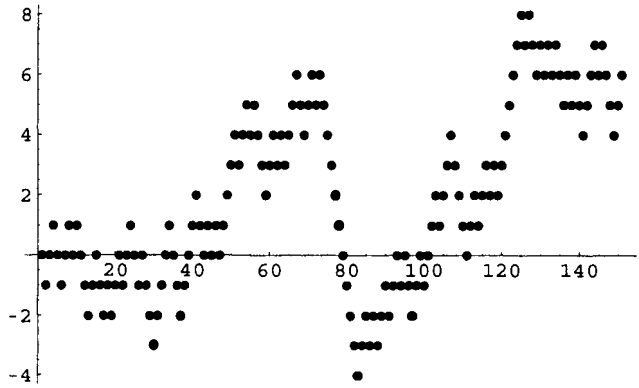


Figure 2.10 Random walk on the integers, with $p = q = 1/2$.

2.7.1 Random Walk on a Graph

As a generalization of Example 2.9, we can associate a random walk with any graph G, whose state space is the vertex set of the graph and whose transition probabilities from i to j are equal to $1/d_i$, where d_i is the degree of i (the number of edges out of i). An important

property of such random walks is that they are time-reversible. This can be easily verified from Kolmogorov's criterion (1.39). In other words, there is no systematic "looping". As a consequence, if the graph is connected and if the stationary distribution $\{\pi_i\}$ exists — which is the case when the graph is finite — then the local balance equations hold:

$$\pi_i p_{ij} = \pi_j p_{ji} \,. \tag{2.39}$$

When $p_{ij} = p_{ji}$ for all i and j, the random walk is said to be *symmetric*. It follows immediately from (2.39) that in this case the equilibrium distribution is uniform over the state space $\mathscr{E}$.

■ **EXAMPLE 2.10 Simple Random Walk on an n-Cube**

We want to simulate a random walk over the vertices of the n-dimensional hypercube (or simply n-cube); see Figure 2.11 for the three-dimensional case.

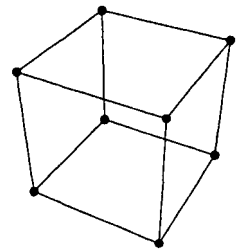


Figure 2.11 At each step, one of the three neighbors of the currently visited vertex is chosen at random.

Note that the vertices of the n-cube are of the form $\mathbf{x} = (x_1, \ldots, x_n)$, with x_i either 0 or 1. The set of all 2^n of these vertices is denoted $\{0,1\}^n$. We generate a random walk $\{X_t, t = 0, 1, 2, \ldots\}$ on $\{0,1\}^n$ as follows. Let the initial state X_0 be arbitrary, say $X_0 = (0, \ldots, 0)$. Given $X_t = (x_{t1}, \ldots, x_{tn})$, choose randomly a coordinate J according to the discrete uniform distribution on the set $\{1, \ldots, n\}$. If j is the outcome, then replace x_{jn} with $1 - x_{jn}$. By doing so we obtain at stage $t+1$

$$X_{t+1} = (x_{t1}, \ldots, 1 - x_{tj}, x_{t(j+1)}, \ldots, x_{tn}) \,,$$

and so on.

2.7.2 Generating Markov Jump Processes

The generation of Markov jump processes is quite similar to the generation of Markov chains above. Suppose $X = \{X_t, t \geqslant 0\}$ is a Markov jump process with transition rates $\{q_{ij}\}$. From Section 1.12.5, recall that the Markov jump process jumps from one state to another according to a Markov chain $Y = \{Y_n\}$ (the jump chain), and the time spent in each state i is exponentially distributed with a parameter that may depend on i. The one-step transition matrix, say K, of Y and the parameters $\{q_i\}$ of the exponential holding times can be found directly from the $\{q_{ij}\}$. Namely, $q_i = \sum_j q_{ij}$ (the sum of the transition rates out of i), and $K(i,j) = q_{ij}/q_i$ for $i \neq j$ (thus, the probabilities are simply proportional to

the rates). Note that $K(i,i) = 0$. Defining the holding times as $A_1, A_2, \ldots$ and the jump times as $T_1, T_2, \ldots$, the algorithm is now as follows.

Algorithm 2.7.2 (Generating a Markov Jump Process)

1. *Initialize* T_0. *Draw* Y_0 *from the initial distribution* $\pi^{(0)}$. *Set* $X_0 = Y_0$. *Set* $n = 0$.
2. *Draw* A_{n+1} *from* $\mathsf{Exp}(q_{Y_n})$.
3. *Set* $T_{n+1} = T_n + A_{n+1}$.
4. *Set* $X_t = Y_n$ *for* $T_n \leqslant t < T_{n+1}$.
5. *Draw* Y_{n+1} *from the distribution corresponding to the* Y_n*-th row of* K, *set* $n = n+1$, *and go to Step 2.*

2.8 GENERATING RANDOM PERMUTATIONS

Many Monte Carlo algorithms involve generating random permutations, that is, random ordering of the numbers $1, 2, \ldots, n$, for some fixed n. For examples of interesting problems associated with the generation of random permutations, see, for instance, the traveling salesman problem in Chapter 6, the permanent problem in Chapter 9, and Example 2.11 below.

Suppose we want to generate each of the $n!$ possible orderings with equal probability. We present two algorithms that achieve this. The first one is based on the ordering of a sequence of n uniform random numbers. In the second, we choose the components of the permutation consecutively. The second algorithm is faster than the first.

Algorithm 2.8.1 (First Algorithm for Generating Random Permutations)

1. *Generate* $U_1, U_2, \ldots, U_n \sim \mathsf{U}(0,1)$ *independently.*
2. *Arrange these in increasing order.*
3. *The indices of the successive ordered values form the desired permutation.*

For example, let $n = 4$ and assume that the generated numbers (U_1, U_2, U_3, U_4) are $(0.7, 0.3, 0.5, 0.4)$. Since $(U_2, U_4, U_3, U_1) = (0.3, 0.4, 0.5, 0.7)$ is the ordered sequence, the resulting permutation is $(2, 4, 3, 1)$. The drawback of this algorithm is that it requires ordering a sequence of n random numbers, which requires $n \ln n$ comparisons.

As we mentioned, the second algorithm is based on the idea of generating the components of the random permutation one by one. The first component is chosen randomly (with equal probability) from $1, \ldots, n$. Next, the second component is randomly chosen from the remaining numbers, and so on. For example, let $n = 4$. We draw component 1 from the discrete uniform distribution on $\{1, 2, 3, 4\}$. Suppose we obtain 2. Our permutation is thus of the form $(2, \cdot, \cdot, \cdot)$. We next generate from the three-point uniform distribution on $\{1, 3, 4\}$. Assume that 1 is chosen. Thus, our intermediate result for the permutation is $(2, 1, \cdot, \cdot)$. Finally, for the third component, choose either 3 or 4 with equal probability. Suppose we draw 4. The resulting permutation is $(2, 1, 4, 3)$. Generating a random variable X from a discrete uniform distribution on $\{x_1, \ldots, x_k\}$ is done efficiently by first generating $I = \lfloor k\,U \rfloor + 1$, with $U \sim \mathsf{U}(0,1)$ and returning $X = x_I$. Thus, we have the following algorithm.

Algorithm 2.8.2 (Second Algorithm for Generating Random Permutations)

1. *Set* $\mathscr{P} = \{1, \ldots, n\}$. *Let* $i = 1$.
2. *Generate* X_i *from the discrete uniform distribution on* $\mathscr{P}$.
3. *Remove* X_i *from* $\mathscr{P}$.
4. *Set* $i = i + 1$. *If* $i \leqslant n$, *go to Step 2.*
5. *Deliver* $(X_1, \ldots, X_n)$ *as the desired permutation.*

Remark 2.8.1 To further improve the efficiency of the second random permutation algorithm, we can implement it as follows: Let $\mathbf{p} = (p_i, \ldots, p_n)$ be a vector that stores the intermediate results of the algorithm at the i-th step. Initially, let $\mathbf{p} = (1, \ldots, n)$. Draw X_1 by uniformly selecting an index $I \in \{1, \ldots, n\}$, and return $X_1 = p_I$. Then *swap* X_1 and $p_n = n$. In the second step, draw X_2 by uniformly selecting I from $\{1, \ldots, n-1\}$, return $X_2 = p_I$ and swap it with p_{n-1}, and so on. In this way, the algorithm requires the generation of only n uniform random numbers (for drawing from $\{1, 2, \ldots, k\}, k = n, n-1, \ldots, 2$) and n swap operations.

■ **EXAMPLE 2.11 Generating a Random Tour in a Graph**

Consider a weighted graph G with n nodes, labeled $1, 2, \ldots, n$. The nodes represent cities, and the edges represent the roads between the cities. The problem is to randomly generate a *tour* that visits all the cities exactly once except for the starting city, which is also the terminating city. Without loss of generality, let us assume that the graph is complete, that is, all cities are connected. We can represent each tour via a permutation of the numbers $1, \ldots, n$. For example, for $n = 4$, the permutation $(1, 3, 2, 4)$ represents the tour $1 \to 3 \to 2 \to 4 \to 1$.

More generally, we represent a tour via a permutation $\mathbf{x} = (x_1, \ldots, x_n)$ with $x_1 = 1$, that is, we assume without loss of generality that we start the tour at city number 1. To generate a random tour uniformly on $\mathscr{X}$, we can simply apply Algorithm 2.8.2. Note that the number of all possible tours of elements in the set of all possible tours $\mathscr{X}$ is

$$|\mathscr{X}| = (n-1)! \tag{2.40}$$

PROBLEMS

2.1 Apply the inverse-transform method to generate a random variable from the discrete uniform distribution with pdf

$$f(x) = \begin{cases} \frac{1}{n+1}, & x = 0, 1, \ldots, n \\ 0 & \text{otherwise.} \end{cases}$$

2.2 Explain how to generate from the Beta$(1, \beta)$ distribution using the inverse-transform method.

2.3 Explain how to generate from the Weib(α, λ) distribution using the inverse-transform method.

2.4 Explain how to generate from the Pareto(α, λ) distribution using the inverse-transform method.

2.5 Many families of distributions are of *location-scale* type. That is, the cdf has the form

$$F(x) = F_0\left(\frac{x-\mu}{\sigma}\right),$$

where μ is called the *location* parameter and σ the *scale* parameter, and F_0 is a fixed cdf that does not depend on μ and σ. The $\mathsf{N}(\mu, \sigma^2)$ family of distributions is a good example, where F_0 is the standard normal cdf. Write $F(x; \mu, \sigma)$ for $F(x)$. Let $X \sim F_0$ (that is, $X \sim F(x; 0, 1)$). Prove that $Y = \mu + \sigma X \sim F(x; \mu, \sigma)$. Thus, to sample from any cdf in a location-scale family, it suffices to know how to sample from F_0.

2.6 Apply the inverse-transform method to generate random variables from a *Laplace distribution* (that is, a shifted two-sided exponential distribution) with pdf

$$f(x) = \frac{\lambda}{2} \mathrm{e}^{-\lambda|x-\theta|}, \quad -\infty < x < \infty, \quad (\lambda > 0).$$

2.7 Apply the inverse-transform method to generate a random variable from the *extreme value distribution*, which has cdf

$$F(x) = 1 - \mathrm{e}^{-\exp\left(\frac{x-\mu}{\sigma}\right)}, \quad -\infty < x < \infty, \quad (\sigma > 0).$$

2.8 Consider the triangular random variable with pdf

$$f(x) = \begin{cases} 0 & \text{if } x < 2a \text{ or } x \geqslant 2b \\ \dfrac{x-2a}{(b-a)^2} & \text{if } 2a \leqslant x < a+b \\ \dfrac{(2b-x)}{(b-a)^2} & \text{if } a+b \leqslant x < 2b \end{cases}$$

a) Derive the corresponding cdf F.

b) Show that applying the inverse-transform method yields

$$X = \begin{cases} 2a + (b-a)\sqrt{2U} & \text{if } 0 \leqslant U < \frac{1}{2} \\ 2b + (a-b)\sqrt{2(1-U)} & \text{if } \frac{1}{2} \leqslant U < 1. \end{cases}$$

2.9 Present an inverse-transform algorithm for generating a random variable from the piecewise-constant pdf

$$f(x) = \begin{cases} C_i, & x_{i-1} \leqslant x \leqslant x_i, \; i = 1, 2, \ldots, n \\ 0 & \text{otherwise,} \end{cases}$$

where $C_i \geqslant 0$ and $x_0 < x_1 < \cdots < x_{n-1} < x_n$.

2.10 Let

$$f(x) = \begin{cases} C_i\, x, & x_{i-1} \leqslant x < x_i,\ \ i = 1, \ldots, n \\ 0 & \text{otherwise,} \end{cases}$$

where $C_i \geqslant 0$ and $x_0 < x_1 < \cdots < x_{n-1} < x_n$.

a) Let $F_i = \sum_{j=1}^{i} \int_{x_{j-1}}^{x_j} C_j\, u\, du$, $i = 1, \ldots, n$. Show that the cdf F satisfies

$$F(x) = F_{i-1} + \frac{C_i}{2}\left(x^2 - x_{i-1}^2\right), \qquad x_{i-1} \leqslant x < x_i,\ \ i = 1, \ldots, n\,.$$

b) Describe an inverse-transform algorithm for random variable generation from $f(x)$.

2.11 A random variable is said to have a *Cauchy* distribution if its pdf is given by

$$f(x) = \frac{1}{\pi}\,\frac{1}{1+x^2}, \quad x \in \mathbb{R}\,. \tag{2.41}$$

Explain how one can generate Cauchy random variables using the inverse-transform method.

2.12 If X and Y are independent standard normal random variables, then $Z = X/Y$ has a Cauchy distribution. Show this. (Hint: first show that if U and $V > 0$ are continuous random variables with joint pdf $f_{U,V}$, then the pdf of $W = U/V$ is given by $f_W(w) = \int_0^\infty f_{U,V}(w\,v, v)\, v\, dv$.)

2.13 Verify the validity of the composition Algorithm 2.3.4.

2.14 Using the composition method, formulate and implement an algorithm for generating random variables from the following normal (Gaussian) mixture pdf:

$$f(x) = \sum_{i=1}^{3} p_i\, \frac{1}{b_i}\, \varphi\left(\frac{x - a_i}{b_i}\right),$$

where φ is the pdf of the standard normal distribution and $(p_1, p_2, p_3) = (1/2, 1/3, 1/6)$, $(a_1, a_2, a_3) = (-1, 0, 1)$, and $(b_1, b_2, b_3) = (1/4, 1, 1/2)$.

2.15 Verify that $C = \sqrt{2\mathrm{e}/\pi}$ in Figure 2.5.

2.16 Prove that if $X \sim \mathsf{Gamma}(\alpha, 1)$, then $X/\lambda \sim \mathsf{Gamma}(\alpha, \lambda)$.

2.17 Let $X \sim \mathsf{Gamma}(1+\alpha, 1)$ and $U \sim \mathsf{U}(0,1)$ be independent. If $\alpha < 1$, then $XU^{1/\alpha} \sim \mathsf{Gamma}(\alpha, 1)$. Prove this.

2.18 If $Y_1 \sim \mathsf{Gamma}(\alpha, 1)$, $Y_2 \sim \mathsf{Gamma}(\beta, 1)$, and Y_1 and Y_2 are independent, then

$$X = \frac{Y_1}{Y_1 + Y_2}$$

is $\mathsf{Beta}(\alpha, \beta)$ distributed. Prove this.

2.19 Devise an acceptance-rejection algorithm for generating a random variable from the pdf f given in (2.20) using an $\mathsf{Exp}(\lambda)$ proposal distribution. Which λ gives the largest acceptance probability?

2.20 The pdf of the truncated exponential distribution with parameter $\lambda = 1$ is given by

$$f(x) = \frac{\mathrm{e}^{-x}}{1 - \mathrm{e}^{-a}}, \quad 0 \leqslant x \leqslant a\,.$$

a) Devise an algorithm for generating random variables from this distribution using the inverse-transform method.

b) Construct a generation algorithm that uses the acceptance-rejection method with an $\mathsf{Exp}(\lambda)$ proposal distribution.

c) Find the efficiency of the acceptance-rejection method for the cases $a = 1$, and a approaching zero and infinity.

2.21 Let the random variable X have pdf

$$f(x) = \begin{cases} \frac{1}{4}, & 0 < x < 1 \\ x - \frac{3}{4}, & 1 \leqslant x \leqslant 2 . \end{cases}$$

Generate a random variable from $f(x)$, using

a) the inverse-transform method,

b) the acceptance-rejection method, using the proposal density

$$g(x) = \frac{1}{2}, \quad 0 \leqslant x \leqslant 2 .$$

2.22 Let the random variable X have pdf

$$f(x) = \begin{cases} \frac{1}{2}x, & 0 < x < 1 \\ \frac{1}{2}, & 1 \leqslant x \leqslant \frac{5}{2} . \end{cases}$$

Generate a random variable from $f(x)$, using

a) the inverse-transform method

b) the acceptance-rejection method, using the proposal density

$$g(x) = \frac{8}{25}x, \quad 0 \leqslant x \leqslant \frac{5}{2} .$$

2.23 Let X have a truncated geometric distribution, with pdf

$$f(x) = cp(1-p)^{x-1}, \; x = 1, \ldots, n ,$$

where c is a normalization constant. Generate a random variable from $f(x)$, using

a) the inverse-transform method,

b) the acceptance-rejection method, with $\mathsf{G}(p)$ as the proposal distribution. Find the efficiency of the acceptance-rejection method for $n = 2$ and $n = \infty$.

2.24 Generate a random variable $Y = \min_{i=1,\ldots,m} \max_{j=1,\ldots,r}\{X_{ij}\}$, assuming that the variables X_{ij}, $i = 1, \ldots, m$, $j = 1, \ldots, r$, are iid with common cdf $F(x)$, using the inverse-transform method. (Hint: use the results for the distribution of order statistics in Example 2.3.)

2.25 Generate 100 $\mathsf{Ber}(0.2)$ random variables three times and produce bar graphs similar to those in Figure 2.6. Repeat for $\mathsf{Ber}(0.5)$.

2.26 Generate a homogeneous Poisson process with rate 100 on the interval $[0, 1]$. Use this to generate a nonhomogeneous Poisson process on the same interval, with rate function

$$\lambda(t) = 100 \sin^2(10t), \quad t \geqslant 0 .$$

2.27 Generate and plot a realization of the points of a two-dimensional Poisson process with rate $\lambda = 2$ on the square $[0, 5] \times [0, 5]$. How many points fall in the square $[1, 3] \times [1, 3]$? How many do you expect to fall in this square?

2.28 Write a program that generates and displays 100 random vectors that are uniformly distributed within the ellipse

$$5x^2 + 21xy + 25y^2 = 9 .$$

2.29 Implement both random permutation algorithms in Section 2.8. Compare their performance.

2.30 Consider a random walk on the undirected graph in Figure 2.12. For example, if the random walk at some time is in state 5, it will jump to 3, 4, or 6 at the next transition, each with probability 1/3.

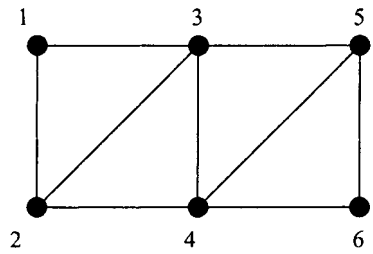


Figure 2.12 A graph.

a) Find the one-step transition matrix for this Markov chain.
b) Show that the stationary distribution is given by $\pi = (\frac{1}{9}, \frac{1}{6}, \frac{2}{9}, \frac{2}{9}, \frac{1}{6}, \frac{1}{9})$.
c) Simulate the random walk on a computer and verify that in the long run, the proportion of visits to the various nodes is in accordance with the stationary distribution.

2.31 Generate various sample paths for the random walk on the integers for $p = 1/2$ and $p = 2/3$.

2.32 Consider the $M/M/1$ queueing system of Example 1.13. Let X_t be the number of customers in the system at time t. Write a computer program to simulate the stochastic process $X = \{X_t\}$ by viewing X as a Markov jump process, and applying Algorithm 2.7.2. Present sample paths of the process for the cases $\lambda = 1$, $\mu = 2$ and $\lambda = 10$, $\mu = 11$.

Further Reading

Classical references on random number generation and random variable generation are [3] and [2]. Other references include [4], [7], and [10] and the tutorial in [9]. A good new reference is [1].

REFERENCES

1. S. Asmussen and P. W. Glynn. *Stochastic Simulation*. Springer-Verlag, New York, 2007.
2. L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.
3. D. E. Knuth. *The Art of Computer Programming*, volume 2: *Seminumerical Algorithms*. Addison-Wesley, Reading, 2nd edition, 1981.
4. A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, 3rd edition, 2000.
5. P. L'Ecuyer. Random numbers for simulation. *Communications of the ACM*, 33(10):85–97, 1990.
6. D. H. Lehmer. Mathematical methods in large-scale computing units. *Annals of the Computation Laboratory of Harvard University*, 26:141–146, 1951.
7. N. N. Madras. *Lectures on Monte Carlo Methods*. American Mathematical Society, 2002.
8. G. Marsaglia and W. Tsang. A simple method for generating gamma variables. *ACM Transactions on Mathematical Software*, 26(3):363–372, 2000.
9. B. D. Ripley. Computer generation of random variables: A tutorial. *International Statistical Review*, 51:301–319, 1983.
10. S. M. Ross. *Simulation*. Academic Press, New York, 3rd edition, 2002.
11. A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3:253–256, 1977.

CHAPTER 3

SIMULATION OF DISCRETE-EVENT SYSTEMS

Computer simulation has long served as an important tool in a wide variety of disciplines: engineering, operations research and management science, statistics, mathematics, physics, economics, biology, medicine, engineering, chemistry, and the social sciences. Through computer simulation, one can study the behavior of real-life systems that are too difficult to examine analytically. Examples can be found in supersonic jet flight, telephone communications systems, wind tunnel testing, large-scale battle management (e.g., to evaluate defensive or offensive weapons systems), or maintenance operations (e.g., to determine the optimal size of repair crews), to mention a few. Recent advances in simulation methodologies, software availability, sensitivity analysis, and stochastic optimization have combined to make simulation one of the most widely accepted and used tools in system analysis and operations research. The sustained growth in size and complexity of emerging real-world systems (e.g., high-speed communication networks and biological systems) will undoubtedly ensure that the popularity of computer simulation continues to grow.

The aim of this chapter is to provide a brief introduction to the art and science of computer simulation, in particular with regard to discrete-event systems. The chapter is organized as follows: Section 3.1 describes basic concepts such as systems, models, simulation, and Monte Carlo methods. Section 3.2 deals with the most fundamental ingredients of discrete-event simulation, namely, the simulation clock and the event list. Finally, in Section 3.3 we further explain the ideas behind discrete-event simulation via a number of worked examples.

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

3.1 SIMULATION MODELS

By a *system* we mean a collection of related entities, sometimes called *components* or *elements*, forming a complex whole. For instance, a hospital may be considered a system, with doctors, nurses, and patients as elements. The elements possess certain characteristics or *attributes* that take on logical or numerical values. In our example, an attribute may be the number of beds, the number of X-ray machines, skill level, and so on. Typically, the activities of individual components interact over time. These activities cause changes in the system's state. For example, the state of a hospital's waiting room might be described by the number of patients waiting for a doctor. When a patient arrives at the hospital or leaves it, the system jumps to a new state.

We shall be solely concerned with *discrete-event systems*, to wit, those systems in which the state variables change instantaneously through jumps at discrete points in time, as opposed to *continuous systems*, where the state variables change continuously with respect to time. Examples of discrete and continuous systems are, respectively, a bank serving customers and a car moving on the freeway. In the former case, the number of waiting customers is a piecewise constant state variable that changes only when either a new customer arrives at the bank or a customer finishes transacting his business and departs from the bank; in the latter case, the car's velocity is a state variable that can change continuously over time.

The first step in studying a system is to build a model from which one can obtain predictions concerning the system's behavior. By a *model* we mean an abstraction of some real system that can be used to obtain predictions and formulate control strategies. Often such models are mathematical (formulas, relations) or graphical in nature. Thus, the actual physical system is translated — through the model — into a mathematical system. In order to be useful, a model must necessarily incorporate elements of two conflicting characteristics: realism and simplicity. On the one hand, the model should provide a reasonably close approximation to the real system and incorporate most of the important aspects of the real system. On the other hand, the model must not be so complex as to preclude its understanding and manipulation.

There are several ways to assess the validity of a model. Usually, we begin testing a model by reexamining the formulation of the problem and uncovering possible flaws. Another check on the validity of a model is to ascertain that all mathematical expressions are dimensionally consistent. A third useful test consists of varying input parameters and checking that the output from the model behaves in a plausible manner. The fourth test is the so-called *retrospective* test. It involves using historical data to reconstruct the past and then determining how well the resulting solution would have performed if it had been used. Comparing the effectiveness of this hypothetical performance with what actually happens then indicates how well the model predicts reality. However, a disadvantage of retrospective testing is that it uses the same data as the model. Unless the past is a representative replica of the future, it is better not to resort to this test at all.

Once a model for the system at hand has been constructed, the next step is to derive a solution from this model. To this end, both *analytical* and *numerical* solutions methods may be invoked. An analytical solution is usually obtained directly from its mathematical representation in the form of formulas. A numerical solution is generally an approximation via a suitable approximation procedure. Much of this book deals with numerical solution and estimation methods obtained via computer simulation. More precisely, we use *stochastic computer simulation* — often called *Monte Carlo simulation* — which includes some randomness in the underlying model, rather than deterministic computer simulation. The

term *Monte Carlo* was used by von Neumann and Ulam during World War II as a code word for secret work at Los Alamos on problems related to the atomic bomb. That work involved simulation of random neutron diffusion in nuclear materials.

Naylor et al. [7] define *simulation* as follows:

> *Simulation is a numerical technique for conducting experiments on a digital computer, which involves certain types of mathematical and logical models that describe the behavior of business or economic systems (or some component thereof) over extended period of real time.*

The following list of typical situations should give the reader some idea of where simulation would be an appropriate tool.

- The system may be so complex that a formulation in terms of a simple mathematical equation may be impossible. Most economic systems fall into this category. For example, it is often virtually impossible to describe the operation of a business firm, an industry, or an economy in terms of a few simple equations. Another class of problems that leads to similar difficulties is that of large-scale, complex queueing systems. Simulation has been an extremely effective tool for dealing with problems of this type.

- Even if a mathematical model can be formulated that captures the behavior of some system of interest, it may not be possible to obtain a solution to the problem embodied in the model by straightforward analytical techniques. Again, economic systems and complex queueing systems exemplify this type of difficulty.

- Simulation may be used as a pedagogical device for teaching both students and practitioners basic skills in systems analysis, statistical analysis, and decision making. Among the disciplines in which simulation has been used successfully for this purpose are business administration, economics, medicine, and law.

- The formal exercise of designing a computer simulation model may be more valuable than the actual simulation itself. The knowledge obtained in designing a simulation study serves to crystallize the analyst's thinking and often suggests changes in the system being simulated. The effects of these changes can then be tested via simulation before implementing them in the real system.

- Simulation can yield valuable insights into the problem of identifying which variables are important and which have negligible effects on the system, and can shed light on how these variables interact; see Chapter 7.

- Simulation can be used to experiment with new scenarios so as to gain insight into system behavior under new circumstances.

- Simulation provides an *in vitro* lab, allowing the analyst to discover better control of the system under study.

- Simulation makes it possible to study dynamic systems in either real, compressed, or expanded time horizons.

- Introducing randomness in a system can actually help solve many optimization and counting problems; see Chapters 6 – 9.

As a modeling methodology, simulation is by no means ideal. Some of its shortcomings and various caveats are: Simulation provides *statistical estimates* rather than *exact* characteristics and performance measures of the model. Thus, simulation results are subject to uncertainty and contain experimental errors. Moreover, simulation modeling is typically time-consuming and consequently expensive in terms of analyst time. Finally, simulation results, no matter how precise, accurate, and impressive, provide consistently useful information about the actual system *only* if the model is a valid representation of the system under study.

3.1.1 Classification of Simulation Models

Computer simulation models can be classified in several ways:

1. *Static versus Dynamic Models.* Static models are those that do not evolve over time and therefore do not represent the passage of time. In contrast, dynamic models represent systems that evolve over time (for example, traffic light operation).

2. *Deterministic versus Stochastic Models.* If a simulation model contains *only* deterministic (i.e., nonrandom) components, it is called *deterministic.* In a deterministic model, all mathematical and logical relationships between elements (variables) are fixed in advance and not subject to uncertainty. A typical example is a complicated and analytically unsolvable system of standard differential equations describing, say, a chemical reaction. In contrast, a model with at least one random input variable is called a *stochastic* model. Most queueing and inventory systems are modeled stochastically.

3. *Continuous versus Discrete Simulation Models.* In discrete simulation models the state variable changes instantaneously at discrete points in time, whereas in continuous simulation models the state changes continuously over time. A mathematical model aiming to calculate a numerical solution for a system of differential equations is an example of continuous simulation, while queueing models are examples of discrete simulation.

This book deals with discrete simulation and in particular with *discrete-event simulation* (DES) models. The associated systems are driven by the occurrence of discrete events, and their state typically changes over time. We shall further distinguish between so-called *discrete-event static systems* (DESS) and *discrete-event dynamic systems* (DEDS). The fundamental difference between DESS and DEDS is that the former do not evolve over time, whereas the latter do. A queueing network is a typical example of a DEDS. A DESS usually involves evaluating (estimating) complex multidimensional integrals or sums via Monte Carlo simulation.

Remark 3.1.1 (Parallel Computing) Recent advances in computer technology have enabled the use of *parallel* or *distributed* simulation, where discrete-event simulation is carried out on multiple linked (networked) computers, operating simultaneously in a cooperative manner. Such an environment allows simultaneous distribution of different computing tasks among the individual processors, thus reducing the overall simulation time.

3.2 SIMULATION CLOCK AND EVENT LIST FOR DEDS

Recall that DEDS evolve over time. In particular, these systems change their state only at a countable number of time points. State changes are triggered by the execution of simulation events occurring at the corresponding time points. Here, an *event* is a collection of attributes (values, types, flags, etc.), chief among which are the *event occurrence time* — or simply *event time* — the *event type*, and an associated algorithm to execute state changes.

Because of their dynamic nature, DEDS require a time-keeping mechanism to advance the simulation time from one event to another as the simulation evolves over time. The mechanism recording the current simulation time is called the *simulation clock*. To keep track of events, the simulation maintains a list of all pending events. This list is called the *event list*, and its task is to maintain all pending events in *chronological* order. That is, events are ordered by their time of occurrence. In particular, the most imminent event is always located at the head of the event list.

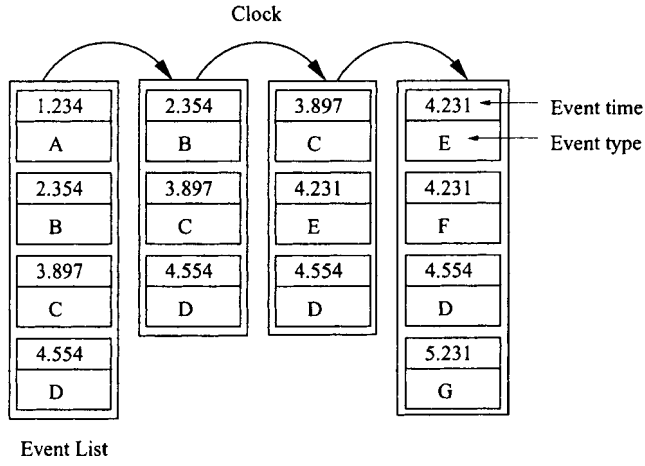


Figure 3.1 The advancement of the simulation clock and event list.

The situation is illustrated in Figure 3.1. The simulation starts by loading the initial events into the event list (chronologically ordered), in this case four events. Next, the most imminent event is unloaded from the event list for execution, and the simulation clock is advanced to its occurrence time, 1.234. After this event is processed and removed, the clock is advanced to the next event, which occurs at time 2.354. In the course of executing a current event, based on its type, the state of the system is updated, and future events are possibly generated and loaded into (or deleted from) the event list. In the above example, the third event — of type C, occurring at time 3.897 — schedules a new event of type E at time 4.231.

The process of unloading events from the event list, advancing the simulation clock, and executing the next most imminent event terminates when some specific stopping condition is met — say, as soon as a prescribed number of customers departs from the system. The following example illustrates this *next-event time advance* approach.

■ **EXAMPLE 3.1**

Money enters a certain bank account in two ways: via frequent small payments and occasional large payments. Suppose that the times between subsequent frequent payments are independent and uniformly distributed on the continuous interval $[7, 10]$ (in days); and, similarly, the times between subsequent occasional payments are independent and uniformly distributed on $[25, 35]$. Each frequent payment is exponentially distributed with a mean of 16 units (say, one unit is \$1000), whereas occasional payments are always of size 100. It is assumed that all payment intervals and sizes are independent. Money is debited from the account at times that form a Poisson process with rate 1 (per day), and the amount debited is normally distributed with mean 5 and standard deviation 1. Suppose that the initial amount of money in the bank account is 150 units.

Note that the state of the system — the account balance — changes only at discrete times. To simulate this DEDS, one need only keep track of when the next frequent and occasional payments occur, as well as the next withdrawal. Denote these three event types by 1, 2, and 3, respectively. We can now implement the event list simply as a 3×2 matrix, where each row contains the event time and the event type. After each advance of the clock, the current event time t and event type i are recorded and the current event is erased. Next, for each event type $i = 1, 2, 3$ the same type of event is scheduled using its corresponding interval distribution. For example, if the event type is 2, then another event of type 2 is scheduled at a time $t + 25 + 10\,U$, where $U \sim \mathsf{U}[0, 1]$. Note that this event can be stored in the same location as the current event that was just erased. However, it is crucial that the event list is then resorted to put the events in chronological order.

A realization of the stochastic process $\{X_t, 0 \leqslant t \leqslant 400\}$, where X_t is the account balance at time t, is given in Figure 3.2, followed by a simple Matlab implementation.

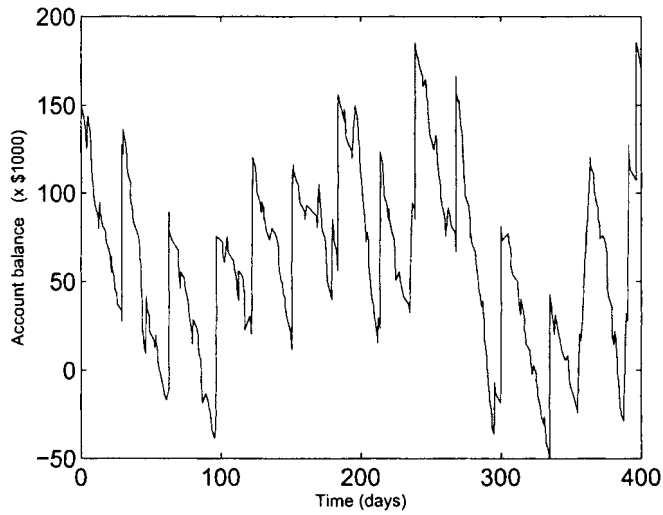


Figure 3.2 A realization of the simulated account balance process.

Matlab Program

```
clear all;
T = 400;
x = 150; %initial amount of money.
xx = [150]; tt = [0];
t=0;
ev_list = inf*ones(3,2);          %record time, type
ev_list(1,:) = [7 + 3*rand, 1];   %schedule type 1 event
ev_list(2,:) = [25 + 10*rand,2]; %schedule type 2 event
ev_list(3,:) = [-log(rand),3];    %schedule type 3 event
ev_list = sortrows(ev_list,1);    % sort event list
while t < T
    t = ev_list(1,1);
    ev_type = ev_list(1,2);
    switch ev_type
        case 1
            x = x + 16*-log(rand);
            ev_list(1,:) = [7 + 3*rand + t, 1];
        case 2
            x = x + 100;
            ev_list(1,:) = [25 + 10*rand + t, 2];
        case 3
            x = x - (5 + randn);
            ev_list(1,:) = [-log(rand) + t, 3];
    end
    ev_list = sortrows(ev_list,1); % sort event list
    xx = [xx,x];
    tt = [tt,t];
end
plot(tt,xx)
```

3.3 DISCRETE-EVENT SIMULATION

As mentioned, DES is the standard framework for the simulation of a large class of models in which the system *state* (one or more quantities that describe the condition of the system) needs to be observed only at certain critical epochs (event times). Between these epochs, the system state either stays the same or changes in a predictable fashion. We further explain the ideas behind DES via two more examples.

3.3.1 Tandem Queue

Figure 3.3 depicts a simple queueing system, consisting of two queues in tandem, called a (Jackson) *tandem queue*. Customers arrive at the first queue according to a Poisson process with rate λ. The service time of a customer at the first queue is exponentially distributed with rate μ_1. Customers who leave the first queue enter the second one. The service time in the second queue has an exponential distribution with rate μ_2. All interarrival and service times are independent.

Suppose we are interested in the number of customers, X_t and Y_t, in the first and second queues, respectively, where we regard a customer who is being served as part of the queue. Figure 3.4 depicts a typical realization of the queue length processes $\{X_t,\ t \geqslant 0\}$ and $\{Y_t,\ t \geqslant 0\}$, obtained via DES.

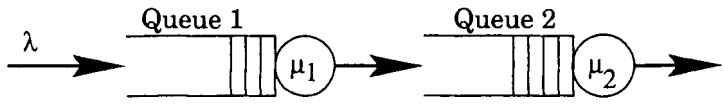


Figure 3.3 A Jackson tandem queue.

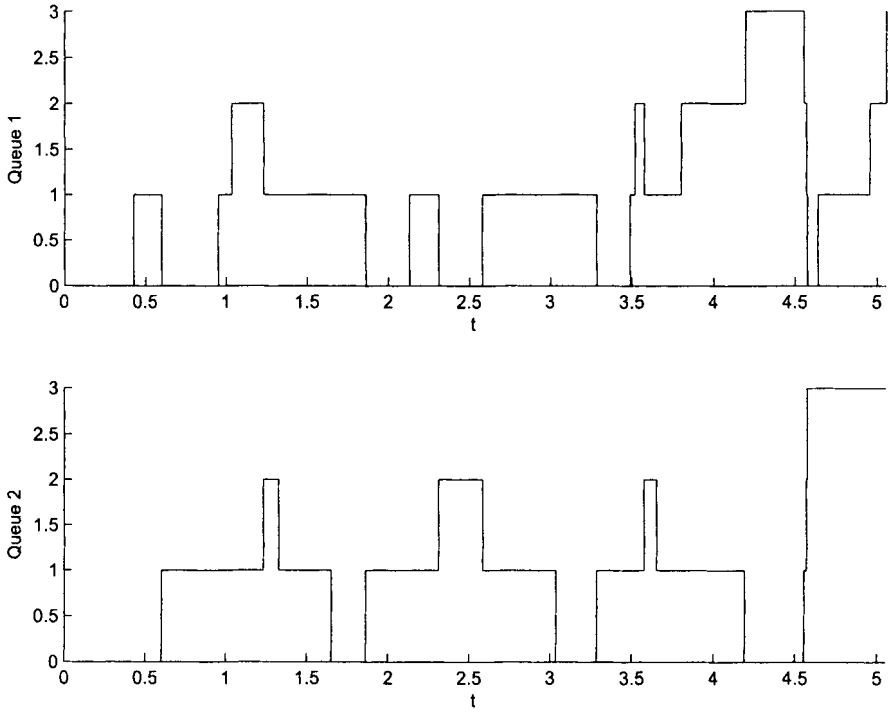


Figure 3.4 A realization of the queue length processes $(X_t, t \geqslant 0)$ and $(Y_t, t \geqslant 0)$.

Before we discuss how to simulate the queue length processes via DES, observe that, indeed, the system evolves via a sequence of discrete events, as illustrated in Figure 3.5. Specifically, the system state (X_t, Y_t) changes only at times of an arrival at the first queue (indicated by A), a departure from the first queue (indicated by D1), and a departure from the second queue (D2).

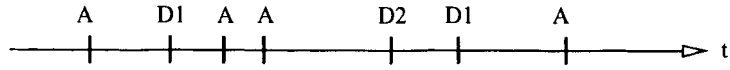


Figure 3.5 A sequence of discrete-events (A = arrival, D1 = departure from the first queue, D2 = departure from the second queue).

There are two fundamental approaches to DES, called the *event-oriented* and *process-oriented* approaches. The pseudocode for an event-oriented implementation of the tandem queue is given in Figure 3.6. The program consists of a main subroutine and separate

subroutines for each event. In addition, the program maintains an ordered list of scheduled current and future events, the so-called *event list*. Each event in the event list has an event *type* ('A', 'D1', and 'D2') and an event *time* (the time at which the arrival or departure will occur). The role of the main subroutine is primarily to progress through the event list and to call the subroutines that are associated with each event type.

Main

```
initialize: Let t = 0, x = 0 and y = 0.
Schedule 'A' at t + Exp(λ).
while TRUE
    Get the first event in the event list
    Let t be the time of this (now current) event
    switch current event type
        case 'A': Call Arrival
        case 'D1': Call Departure1
        case 'D2': Call Departure2
    end
    Remove the current event from the event list and sort the event list.
end
```

Figure 3.6 Main subroutine of an event-oriented simulation program.

The role of the event subroutines is to update the system state and to schedule new events into the event list. For example, an arrival event at time t will trigger another arrival event at time $t + Z$, with $Z \sim \text{Exp}(\lambda)$. We write this, as in the Main routine, in shorthand as $t + \text{Exp}(\lambda)$. Moreover, if the first queue is empty, it will also trigger a departure event from the first queue at time $t + \text{Exp}(\mu_1)$.

Arrival

```
Schedule 'A'
    at t + Exp(λ)
if x = 0
    Schedule 'D1'
        at t + Exp(μ1)
end
x = x + 1
```

Departure1

```
x = x − 1
if x ≠ 0
    Schedule 'D1'
        at t + Exp(μ1)
end
if x = 0
    Schedule 'D2'
        at t + Exp(μ2)
end
y = y + 1
```

Departure2

```
y = y − 1
if y ≠ 0
    Schedule 'D2'
        at t + Exp(μ2)
end
```

Figure 3.7 Event subroutines of an event-oriented simulation program.

The process-oriented approach to DES is much more flexible than the event-oriented approach. A process-oriented simulation program closely resembles the actual processes that drive the simulation. Such simulation programs are invariably written in an object-oriented programming language, such as Java or C++. We illustrate the process-oriented approach via our tandem queue example. In contrast to the event-oriented approach, customers,

servers, and queues are now actual entities, or *objects* in the program, that can be manipulated. The queues are passive objects that can contain various customers (or be empty), and the customers themselves can contain information such as their arrival and departure times. The servers, however, are active objects (*processes*), which can interact with each other and with the passive objects. For example, the first server takes a client out of the first queue, serves the client, and puts her into the second queue when finished, alerting the second server that a new customer has arrived if necessary. To generate the arrivals, we define a *generator* process that generates a client, puts it in the first queue, alerts the first server if necessary, holds for a random interarrival time (we assume that the interarrival times are iid), and then repeats these actions to generate the next client.

As in the event-oriented approach, there exists an event list that keeps track of the current and pending events. However, this event list now contains *processes*. The process at the top of the event list is the one that is currently active. Processes may ACTIVATE other processes by putting them at the head of the event list. Active processes may HOLD their action for a certain amount of time (such processes are put further up in the event list). Processes may PASSIVATE altogether (temporarily remove themselves from the event list). Figure 3.8 lists the typical structure of a process-oriented simulation program for the tandem queue.

Main

```
1: initialize: create the two queues, the two servers and the generator
2: ACTIVATE the generator
3: HOLD(duration of simulation)
4: STOP
```

Generator

```
1: while TRUE
2:     generate new client
3:     put client in the first queue
4:     if server 1 is idle
5:         ACTIVATE server 1
6:     end
7:     HOLD(interarrival time)
8: end
```

Server 1

```
1: while TRUE
2:     if waiting room is empty
3:         PASSIVATE
4:     else
5:         get first customer from waiting
               room
6:         HOLD(service time)
7:     end
8:     put customer in queue 2
9:     if server 2 is idle
10:        ACTIVATE server 2
11:    end
12: end
```

Figure 3.8 The structure of a process-oriented simulation program for the tandem queue. The Server 2 process is similar to the Server 1 process, with lines 8–11 replaced with "remove customer from system".

The collection of statistics, for example the waiting time or queue length, can be done by different objects and at various stages in the simulation. For example, customers can record their arrival and departure times and report or record them just before they leave the system. There are many freely available object-oriented simulation environments nowadays, such as SSJ, J-Sim, and C++Sim, all inspired by the pioneering simulation language SIMULA.

3.3.2 Repairman Problem

Imagine n machines working simultaneously. The machines are unreliable and fail from time to time. There are $m < n$ identical repairmen, who can each work only on one machine at a time. When a machine has been repaired, it is as good as new. Each machine has a fixed lifetime distribution and repair time distribution. We assume that the lifetimes and repair times are independent of each other. Since the number of repairmen is less than the number of machines, it can happen that a machine fails and all repairmen are busy repairing other failed machines. In that case, the failed machine is placed in a queue to be served by the next available repairman. When upon completion of a repair job a repairman finds the failed machine queue empty, he enters the repair pool and remains idle until his service is required again. We assume that machines and repairmen enter their respective queues in a first-in-first-out (FIFO) manner. The system is illustrated in Figure 3.9 for the case of three repairmen and five machines.

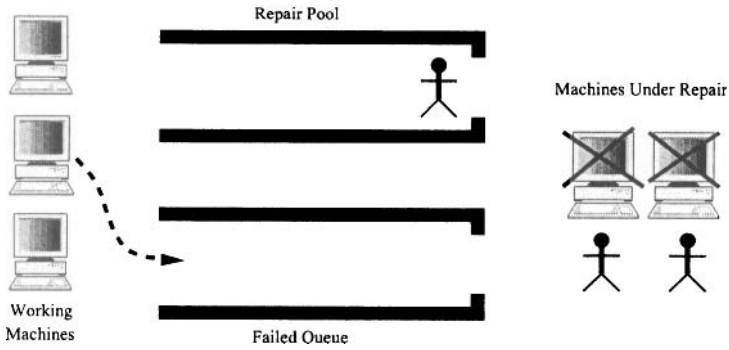


Figure 3.9 The repairman system.

For this particular model the system state could be comprised of the number of available repairmen R_t and the number of failed machines F_t at any time t. In general, the stochastic process $\{(F_t, R_t), t \geqslant 0\}$ is not a Markov process unless the service and lifetimes have exponential distributions.

As with the tandem queue, we first describe an event-oriented and then a process-oriented approach for this model.

3.3.2.1 Event-Oriented Approach There are two types of events: failure events 'F' and repair events 'R'. Each event triggers the execution of the corresponding failure or repair procedure. The task of the main program is to advance the simulation clock and to assign the correct procedure to each event. Denoting n_f the number of failed machines and n_r the number of available repairmen, the main program is thus of the following form:

MAIN PROGRAM

```
1: initialize: Let t = 0, n_r = m and n_f = 0
2: for i = 1 to n
3:     Schedule 'F' of machine i at time t+lifetime(i)
4: end
5: while TRUE
6:     Get the first event in the event list
7:     Let t be the time of this (now current) event
8:     Let i be the machine number associated with this event
9:     switch current event type
10:        case 'F': Call Failure
11:        case 'R': Call Repair
12:    end
13:    Remove the current event from the event list
14: end
```

Upon failure, a repair needs to be scheduled at a time equal to the current time plus the required repair time for this machine. However, this is true only if there is a repairman available to carry out the repairs. If this is not the case, the machine is placed in the "failed" queue. The number of failed machines is always increased by 1. The failure procedure is thus as follows:

FAILURE PROCEDURE

```
1: if (n_r > 0)
2:     Schedule 'R' of machine i at time t+repairtime(i)
3:     n_r = n_r - 1
4: else Add the machine to the repair queue
5: end
6: n_f = n_f + 1
```

Upon repair, the number of failed machines is decreased by 1. The machine that has just been repaired is scheduled for a failure after the lifetime of the machine. If the "failed" queue is not empty, the repairman takes the next machine from the queue and schedules a corresponding repair event. Otherwise, the number of idle/available repairmen is increased by 1. This gives the following repair procedure:

REPAIR PROCEDURE

```
1: n_f = n_f - 1
2: Schedule 'F' for machine i at time t+lifetime(i)
3: if repair pool not empty
4:     Remove the first machine from the "failed" queue; let j be its number
5:     Schedule 'R' of machine j at time t+repairtime(j)
6: else n_r = n_r + 1
7: end
```

3.3.2.2 Process-Oriented Approach To outline a process-oriented approach for any simulation it is convenient to represent the processes by flowcharts. In this case there are two processes: the repairman process and the machine process. The flowcharts in Figure 3.10 are self-explanatory. Note that the horizontal parallel lines in the flowcharts indicate that the process PASSIVATEs, that is, the process temporarily stops (is removed from the event list), until it is ACTIVATEd by another process. The circled letters A and B indicate how the two interact. A cross in the flowchart indicates that the process is rescheduled in the event list (E.L.). This happens in particular when the process HOLDs for an amount of time. After holding it resumes from where it left off.

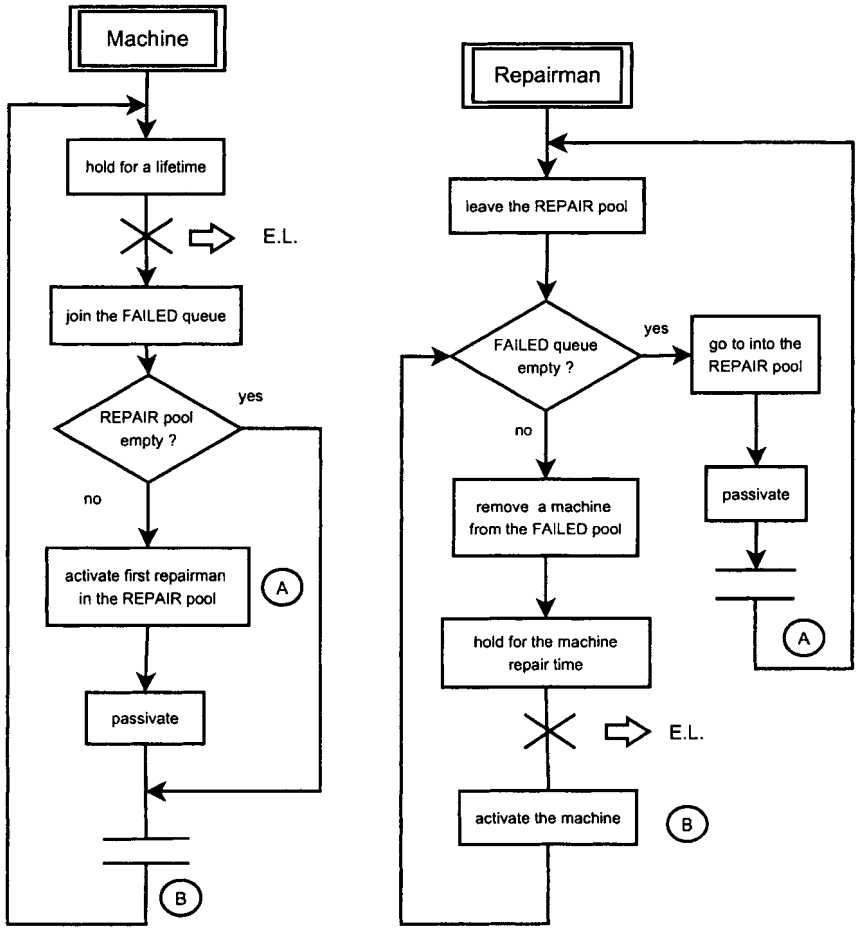


Figure 3.10 Flowcharts for the two processes in the repairman problem.

PROBLEMS

3.1 Consider the $M/M/1$ queueing system in Example 1.13. Let X_t be the number of customers in the system at time t. Write a computer program to simulate the stochastic process $X = \{X_t, t \geqslant 0\}$ using an event- or process-oriented DES approach. Present sample paths of the process for the cases $\lambda = 1$, $\mu = 2$ and $\lambda = 10$, $\mu = 11$.

3.2 Repeat the above simulation, but now assume $\mathrm{U}(0, 2)$ interarrival times and $\mathrm{U}(0, 1/2)$ service times (all independent).

3.3 Run the Matlab program of Example 3.1 (or implement it in the computer language of your choice). Out of 1000 runs, how many lead to a negative account balance during the first 100 days? How does the process behave for large t?

3.4 Implement an event-oriented simulation program for the tandem queue. Let the interarrivals be exponentially distributed with mean 5, and let the service times be uniformly distributed on [3,6]. Plot realizations of the queue length processes of both queues.

3.5 Consider the repairman problem with two identical machines and one repairman. We assume that the lifetime of a machine has an exponential distribution with expectation 5 and that the repair time of a machine is exponential with expectation 1. All the lifetimes and repair times are independent of each other. Let X_t be the number of failed machines at time t.

- **a)** Verify that $X = \{X_t, t \geqslant 0\}$ is a birth-and-death process, and give the corresponding birth and death rates.
- **b)** Write a program that simulates the process X according to Algorithm 2.7.2 and use this to assess the fraction of time that both machines are out of order. Simulate from $t = 0$ to $t = 100{,}000$.
- **c)** Write an event-oriented simulation program for this process.
- **d)** Let the exponential life and repair times be uniformly distributed, on $[0, 10]$ and $[0, 2]$, respectively (hence the expectations stay the same as before). Simulate from $t = 0$ to $t = 100{,}000$. How does the fraction of time that both machines are out of order change?
- **e)** Now simulate a repairman problem with the above life and repair times, but now with five machines and three repairmen. Run again from $t = 0$ to $t = 100{,}000$.

3.6 Draw flow diagrams, such as in Figure 3.10, for all the processes in the tandem queue; see also Figure 3.8.

3.7 Consider the following queueing system. Customers arrive at a circle, according to a Poisson process with rate λ. On the circle, which has circumference 1, a single server travels at constant speed α^{-1}. Upon arrival the customers choose their positions on the circle according to a uniform distribution. The server always moves toward the nearest customer, sometimes clockwise, sometimes counterclockwise. Upon reaching a customer, the server stops and serves the customer according to an exponential service time distribution with parameter μ. When the server is finished, the customer is removed from the circle and the server resumes his journey on the circle. Let $\eta = \lambda\alpha$, and let $X_t \in [0, 1]$ be the position of the server at time t. Furthermore, let N_t be the number of customers waiting on the circle at time t. Implement a simulation program for this so-called *continuous poling system with a "greedy" server*, and plot realizations of the processes $\{X_t, t \geqslant 0\}$ and $\{N_t, t \geqslant 0\}$, taking the parameters $\lambda = 1$, $\mu = 2$, for different values of α. Note that although the state

space of $\{X_t, t \geqslant 0\}$ is continuous, the system is still a DEDS since between arrival and service events the system state changes deterministically.

3.8 Consider a *continuous flow line* consisting of three machines in tandem separated by two storage areas, or buffers, through which a continuous (fluid) stream of items flows from one machine to the next; see Figure 3.11.

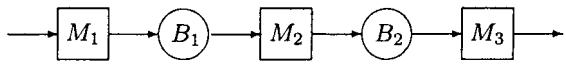


Figure 3.11 A flow line with three machines and two buffers (three-stage flow line).

Each machine $i = 1, 2, 3$ has a specific *machine speed* ν_i, which is the maximum rate at which it can transfer products from its upstream buffer to its downstream buffer. The lifetime of machine i has an exponential distribution with parameter λ_i. The repair of machine i starts immediately after failure and requires an exponential time with parameter μ_i. All life and repair times are assumed to be independent of each other. Failures are operation independent. In particular, the failure rate of a "starved" machine (a machine that is idle because it does not receive input from its upstream buffer) is the same as that of a fully operational machine. The first machine has an unlimited supply.

Suppose all machine speeds are 1, the buffers are of equal size b, and all machines are identical with parameters $\lambda = 1$ and $\mu = 2$.

a) Implement an event- or process-oriented simulation program for this system.

b) Assess via simulation the average *throughput* of the system (the long-run amount of fluid that enters/leaves the system per unit of time) as a function of the buffer size b.

Further Reading

One of the first books on Monte Carlo simulation is by Hammersley and Handscomb [3]. Kalos and Whitlock [4] is another classical reference. The event- and process-oriented approaches to discrete-event simulation are elegantly explained in Mitrani [6]. Among the great variety of books on DES, all focusing on different aspects of the modeling and simulation process, we mention [5], [8], [1], and [2]. The choice of computer language in which to implement a simulation program is very subjective. The simple models discussed in this chapter can be implemented in any standard computer language, even Matlab, although the latter does not provide easy event list manipulation. Commercial simulation environments such as ARENA/SIMAN and SIMSCRIPT II.5 make the implementation of larger models much easier. Alternatively, various free SIMULA-like Java packages exist that offer fast implementation of event- and process-oriented simulation programs. Examples are Pierre L'Ecuyer's SSJ `http://www.iro.umontreal.ca/~simardr/ssj/`, DSOL `http://sk-3.tbm.tudelft.nl/simulation/`, developed by the Technical University Delft, and J-SIM `http://www.j-sim.zcu.cz/`.

REFERENCES

1. J. S. Banks, J. S. Carson II, B. L. Nelson, and D. M. Nicol. *Discrete-Event System Simulation.* Prentice-Hall, Englewood Cliffs, NJ, 4th edition, 2004.
2. G. S. Fishman. *Discrete Event Simulation: Modeling, Programming, and Analysis.* Springer-Verlag, New York, 2001.
3. J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods.* John Wiley & Sons, New York, 1964.
4. M. H. Kalos and P. A. Whitlock. *Monte Carlo Methods, Volume I: Basics.* John Wiley & Sons, New York, 1986.
5. A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, New York, 3rd edition, 2000.
6. I. Mitrani. *Simulation Techniques for Discrete Event Systems.* Cambridge University Press, Cambridge, 1982.
7. T. J. Naylor, J. L. Balintfy, D. S. Burdick, and K. Chu. *Computer Simulation Techniques.* John Wiley & Sons, New York, 1966.
8. R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling.* John Wiley & Sons, New York, 1998.

CHAPTER 4

STATISTICAL ANALYSIS OF DISCRETE-EVENT SYSTEMS

4.1 INTRODUCTION

An essential part of a simulation study is the statistical analysis of the output data, that is, the data obtained from the simulation model. In this chapter we present several important statistical techniques applied to different types of simulation models. As explained in the previous chapter, simulation models can generally be divided into *static* and *dynamic* models. In both cases the behavior of the system is described by the *system state* which, for all practical purposes, can be thought of as a finite-dimensional random vector $\mathbf{X}$ containing all the information about the system. In static models the system state does not depend on time. The simulation of such models involves the repeated generation of the system state, and can be implemented using the algorithms in Chapter 2. In dynamic models the system state *does* depend on time, for example, $\mathbf{X}_t$ at time t. The behavior of the system is described by a discrete- or continuous-time stochastic process $\{\mathbf{X}_t\}$.

The rest of this chapter is organized as follows. Section 4.2 treats the statistical analysis of the output data from static models. Section 4.3 discusses the difference between finite-horizon and steady-state simulation for dynamic models. In Section 4.3.2 we consider steady-state simulation in more detail. Two popular methods for estimating steady-state performance measures — the batch means and regenerative methods — are discussed in Sections 4.3.2.1 and 4.3.2.2, respectively. Finally, in Section 4.4 we present the bootstrap technique.

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

4.2 STATIC SIMULATION MODELS

As mentioned in Chapter 3, in a static simulation model the system state does not depend on time. Suppose we want to determine the expectation

$$\ell = \mathbb{E}[H(\mathbf{X})] = \int H(\mathbf{x})\, f(\mathbf{x})\, \mathrm{d}\mathbf{x} \,, \tag{4.1}$$

where $\mathbf{X}$ is a random vector with pdf f, and $H(\mathbf{x})$ is a real-valued function called the *performance* function. We assume that ℓ cannot be evaluated analytically and we need to resort to simulation. An unbiased estimator of ℓ is the *sample mean*

$$\widehat{\ell} = N^{-1} \sum_{i=1}^{N} H(\mathbf{X}_i) \,, \tag{4.2}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a *random sample* from f, that is, the $\{\mathbf{X}_i\}$ are independent replications of $\mathbf{X} \sim f$.

■ EXAMPLE 4.1 Reliability Model

Consider a system that consists of n components. The operational state of each component $i = 1, \ldots, n$ is represented by $X_i \sim \mathrm{Ber}(p_i)$, where $X_i = 1$ means that the component is working and $X_i = 0$ means that it has failed. Note that the probability that component i is working — its *reliability* — is p_i. The failure behavior of the system is thus represented by the binary random vector $\mathbf{X} = (X_1, \ldots, X_n)$, where it is usually assumed that the $\{X_i\}$ are independent. Suppose that the operational state of the system, say Y, is either functioning or failed, depending on the operational states of the components. In other words, we assume that there exists a function $H : \mathscr{X} \to \{0, 1\}$ such that

$$Y = H(\mathbf{X}) \,,$$

where $\mathscr{X} = \{0, 1\}^n$ is the set of all binary vectors of length n.

The function H is called the *structure function* and often can be represented by a graph. In particular, the graph in Figure 4.1 depicts a *bridge network* with five components (links). For this particular model the system works (that is, $H(\mathbf{X}) = 1$) if the black terminal nodes are connected by working links. The structure function is equal to (see Problem 4.2)

$$H(\mathbf{x}) = 1 - (1 - x_1\, x_4)\,(1 - x_2\, x_5)\,(1 - x_1\, x_3\, x_5)\,(1 - x_2\, x_3\, x_4) \,. \tag{4.3}$$

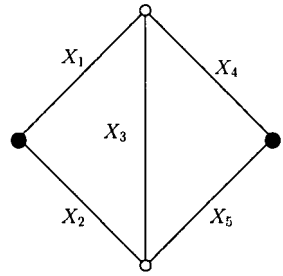


Figure 4.1 A bridge network.

Suppose we are interested in the reliability ℓ of the general n-component system. We have

$$\begin{aligned} \ell = \mathbb{P}(Y = 1) = \mathbb{E}[H(\mathbf{X})] &= \sum_{\mathbf{x} \in \mathscr{X}} H(\mathbf{x})\, \mathbb{P}(\mathbf{X} = \mathbf{x}) \\ &= \sum_{\mathbf{x} \in \mathscr{X}} H(\mathbf{x}) \prod_{i=1}^{n} \left[p_i^{x_i} (1 - p_i)^{1 - x_i} \right] . \end{aligned} \tag{4.4}$$

For complex systems with a large number of components and with little structure, it is very time-consuming to compute the system reliability ℓ via (4.4), since this requires the evaluation of $\mathbb{P}(\mathbf{X} = \mathbf{x})$ and $H(\mathbf{x})$ for 2^n vectors $\mathbf{x}$. However, simulation of $\mathbf{X}$ and estimation of ℓ via (4.2) can still be a viable approach, even for large systems, provided that $H(\mathbf{X})$ is readily evaluated. In practice one needs substantially fewer than 2^n samples to estimate ℓ accurately.

■ EXAMPLE 4.2 Stochastic PERT Network

The *program evaluation and review technique* (PERT) is a frequently used tool for project management. Typically a project consists of many activities, some of which can be performed in parallel, while others can only be performed after certain preceding activities have been finished. In particular, each activity has a list of *predecessors*, which must be completed before it can start. A PERT network is a directed graph where the arcs represent the activities and the vertices represent specific milestones. A milestone is completed when all activities pointing to that milestone are completed. Before an activity can begin, the milestone from which the activity originates must be completed. An example of a precedence list of activities is given in Table 4.2; its PERT graph is given in Figure 4.2.

Table 4.1 Precedence ordering of activities.

Activity	1	2	3	4	5	6	7	8	9	10	11	12
Predecessor(s)	-	-	1	1	2	2	3	3	4,6	5,8	7	9,10

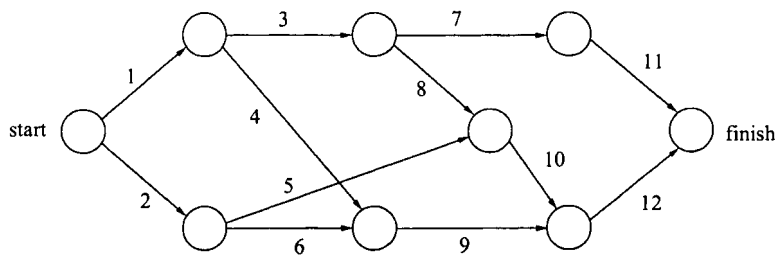


Figure 4.2 A stochastic PERT network.

Suppose each activity i will take a random time X_i to complete. An important quantity for PERT networks is the maximal project duration, that is, the length of the longest path from start to finish — the so-called *critical path*. Suppose we are

interested in the expected maximal project duration, say ℓ. Letting $\mathbf{X}$ be the vector of activity lengths and $H(\mathbf{X})$ be the length of the critical path, we have

$$\ell = \mathbb{E}[H(\mathbf{X})] = \mathbb{E}\left[\max_{j=1,\ldots,p} \sum_{i \in \mathscr{P}_j} X_i\right], \tag{4.5}$$

where $\mathscr{P}_j$ is the j-th complete path from start to finish and p is the number of such paths.

4.2.1 Confidence Interval

In order to specify how *accurate* a particular estimate $\widehat{\ell}$ is, that is, how close it is to the actual unknown parameter ℓ, one needs to provide not only a point estimate $\widehat{\ell}$ but a confidence interval as well. To do so, recall from Section 1.13 that by the central limit theorem $\widehat{\ell}$ has approximately a $\mathsf{N}(\ell, \sigma^2/N)$ distribution, where σ^2 is the variance of $H(\mathbf{X})$. Usually σ^2 is unknown, but it can be estimated with the *sample variance*

$$S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (H(\mathbf{X}_i) - \widehat{\ell})^2 , \tag{4.6}$$

which (by the law of large numbers) tends to σ^2 as $N \to \infty$. Consequently, for large N we see that $\widehat{\ell}$ is approximately $\mathsf{N}(\ell, S^2/N)$ distributed. Thus, if z_γ denotes the γ-quantile of the $\mathsf{N}(0,1)$ distribution (this is the number such that $\Phi(z_\gamma) = \gamma$, where Φ denotes the standard normal cdf; for example $z_{0.95} = 1.645$, since $\Phi(1.645) = 0.95$), then

$$\mathbb{P}\left(\widehat{\ell} - z_{1-\alpha/2}\frac{S}{\sqrt{N}} \leqslant \ell \leqslant \widehat{\ell} + z_{1-\alpha/2}\frac{S}{\sqrt{N}}\right) \approx 1 - \alpha .$$

In other words, an approximate $(1-\alpha)100\%$ *confidence interval* for ℓ is

$$\left(\widehat{\ell} \pm z_{1-\alpha/2}\frac{S}{\sqrt{N}}\right), \tag{4.7}$$

where the notation $(a \pm b)$ is shorthand for the interval $(a-b, a+b)$.

It is common practice in simulation to use and report the *absolute and relative* widths of this confidence interval, defined as

$$w_a = 2\, z_{1-\alpha/2}\frac{S}{\sqrt{N}} \tag{4.8}$$

and

$$w_r = \frac{w_a}{\widehat{\ell}}, \tag{4.9}$$

respectively, provided that $\widehat{\ell} > 0$. The absolute and relative widths may be used as stopping rules (criteria) to control the length of a simulation run. The relative width is particularly useful when ℓ is very small. For example, think of ℓ as the unreliability (1 minus the reliability) of a system in which all the components are very reliable. In such a case ℓ could be as small as $\ell \approx 10^{-10}$, so that reporting a result such as $w_a = 0.05$ is almost meaningless,

while in contrast, reporting $w_r = 0.05$ is quite meaningful. Another important quantity is the *relative error* (RE) of the estimator $\widehat{\ell}$, defined (see also (1.47)) as

$$\mathrm{RE} = \frac{\sqrt{\mathrm{Var}(\widehat{\ell})}}{\mathbb{E}[\widehat{\ell}]} = \frac{\sigma}{\ell\sqrt{N}}\,, \tag{4.10}$$

which can be estimated as $S/(\widehat{\ell}\sqrt{N})$. Note that this is equal to w_r divided by $2z_{1-\alpha/2}$.

The following algorithm summarizes how to estimate the expected system performance, $\ell = \mathbb{E}[H(\mathbf{X})]$, and how to calculate the corresponding confidence interval.

Algorithm 4.2.1

1. *Perform N replications, $\mathbf{X}_1, \ldots, \mathbf{X}_N$, for the underlying model and calculate $H(\mathbf{X}_i)$, $i = 1, \ldots, N$.*
2. *Calculate a point estimate and a confidence interval of ℓ from (4.2) and (4.7), respectively.*

4.3 DYNAMIC SIMULATION MODELS

Dynamic simulation models deal with systems that evolve over time. Our goal is (as for static models) to estimate the expected system performance, where the state of the system is now described by a stochastic process $\{\mathbf{X}_t\}$, which may have a continuous or discrete time parameter. For simplicity we mainly consider the case where $\mathbf{X}_t$ is a scalar random variable; we then write X_t instead of $\mathbf{X}_t$.

We make a distinction between *finite-horizon* and *steady-state* simulation. In finite-horizon simulation, measurements of system performance are defined relative to a specified interval of simulation time $[0, T]$ (where T may be a random variable), while in steady-state simulation, performance measures are defined in terms of certain limiting measures as the time horizon (simulation length) goes to infinity.

The following illustrative example offers further insight into finite-horizon and steady-state simulation. Suppose that the state X_t represents the number of customers in a stable $M/M/1$ queue (see Example 1.13 on page 26). Let

$$F_{t,m}(x) = \mathbb{P}(X_t \leqslant x \,|\, X_0 = m) \tag{4.11}$$

be the cdf of X_t given the initial state $X_0 = m$ (m customers are initially present). $F_{t,m}$ is called the *finite-horizon distribution* of X_t given that $X_0 = m$.

We say that the process $\{X_t\}$ *settles into steady-state* (equivalently, that *steady-state exists*) if for all m

$$\lim_{t\to\infty} F_{t,m}(x) = F(x) \equiv \mathbb{P}(X \leqslant x) \tag{4.12}$$

for some random variable X. In other words, *steady-state* implies that, as $t \to \infty$, the transient cdf, $F_{t,m}(x)$ (which generally depends on t and m), approaches a steady-state cdf, $F(x)$, which *does not depend* on the initial state, m. The stochastic process, $\{X_t\}$, is said to *converge in distribution* to a random variable $X \sim F$. Such an X can be interpreted as the random state of the system when observed far away in the future. The operational meaning of *steady-state* is that after some period of time the transient cdf $F_{t,m}(x)$ comes close to its limiting (steady-state) cdf $F(x)$. It is important to realize that this does *not* mean

that at any point in time the realizations of $\{X_t\}$ generated from the simulation run become independent or constant. The situation is illustrated in Figure 4.3, where the dashed curve indicates the expectation of X_t.

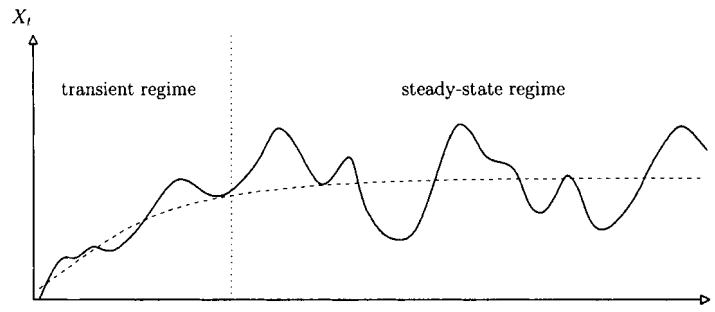


Figure 4.3 The state process for a dynamic simulation model.

The exact distributions (transient and steady-state) are usually available only for simple Markovian models such as the $M/M/1$ queue. For non-Markovian models, usually neither the distributions (transient and steady-state) nor even the associated moments are available via analytical methods. For performance analysis of such models one must resort to simulation.

Note that for some stochastic models, only finite-horizon simulation is feasible, since the steady-state regime either does not exist or the finite-horizon period is so long that the steady-state analysis is computationally prohibitive (see, for example, [9]).

4.3.1 Finite-Horizon Simulation

The statistical analysis for finite-horizon simulation models is basically the same as that for static models. To illustrate the procedure, suppose that $\{X_t, t \geqslant 0\}$ is a continuous-time process for which we wish to estimate the expected average value,

$$\ell(T, m) = \mathbb{E}\left[T^{-1}\int_0^T X_t \, \mathrm{d}t\right], \tag{4.13}$$

as a function of the time horizon T and the initial state $X_0 = m$. (For a discrete-time process $\{X_t, t = 1, 2, \ldots\}$ the integral $\int_0^T X_t \, dt$ is replaced by the sum $\sum_{t=1}^{T} X_t$.) As an example, if X_t represents the number of customers in a queueing system at time t, then $\ell(T, m)$ is the average number of customers in the system during the time interval $[0, T]$, given $X_0 = m$.

Assume now that N independent replications are performed, each starting at state $X_0 = m$. Then the point estimator and the $(1 - \alpha)$ 100% confidence interval for $\ell(T, m)$ can be written, as in the static case (see (4.2) and (4.7)) , as

$$\widehat{\ell}(T, m) = N^{-1}\sum_{i=1}^{N} Y_i \tag{4.14}$$

and

$$\left(\widehat{\ell}(T, m) \pm z_{1-\alpha/2} S N^{-1/2}\right), \tag{4.15}$$

respectively, where $Y_i = T^{-1} \int_0^T X_{ti}\, dt$, X_{ti} is the observation at time t from the i-th replication and S^2 is the sample variance of $\{Y_i\}$. The algorithm for estimating the finite-horizon performance, $\ell(T, m)$, is thus:

Algorithm 4.3.1

1. *Perform N independent replications of the process $\{X_t, t \leqslant T\}$, starting each replication from the initial state $X_0 = m$.*
2. *Calculate the point estimator and the confidence interval of $\ell(T, m)$ from (4.14) and (4.15), respectively.*

If, instead of the expected average number of customers, we want to estimate the expected *maximum* number of customers in the system during an interval $(0, T]$, the only change required is to replace $Y_i = T^{-1} \int_0^T X_{ti}\, dt$ with $Y_i = \max_{0 \leqslant t \leqslant T} X_{ti}$. In the same way, we can estimate other performance measures for this system, such as the probability that the maximum number of customers during $(0, T]$ exceeds some level γ or the expected average period of time that the first k customers spend in the system.

4.3.2 Steady-State Simulation

Steady-state simulation concerns systems that exhibit some form of stationary or long-run behavior. Loosely speaking, we view the system as having started in the infinite past, so that any information about initial conditions and starting times becomes irrelevant. The more precise notion is that the system state is described by a *stationary process*; see also Section 1.12.

■ **EXAMPLE 4.3** $M/M/1$ **Queue**

Consider the birth and death process $\{X_t, t \geqslant 0\}$ describing the number of customers in the $M/M/1$ queue; see Example 1.13. When the traffic intensity $\varrho = \lambda/\mu$ is less than 1, this Markov jump process has a limiting distribution,

$$\lim_{t \to \infty} \mathbb{P}(X_t = k) = (1 - \varrho)\varrho^k, \quad k = 0, 1, 2, \ldots,$$

which is also its stationary distribution. When X_0 is distributed according to this limiting distribution, the process $\{X_t, t \geqslant 0\}$ is stationary: it behaves as if it has been going on for an infinite period of time. In particular, the distribution of X_t does not depend on t. A similar result holds for the Markov process $\{Z_n, n = 1, 2, \ldots\}$, describing the number of customers in the system as seen by the n-th arriving customer. It can be shown that under the condition $\varrho < 1$ it has the *same* limiting distribution as $\{X_t, t \geqslant 0\}$. Note that for the $M/M/1$ queue the steady-state expected performance measures are available analytically, while for the $GI/G/1$ queue, to be discussed in Example 4.4, one needs to resort to simulation.

Special care must be taken when making inferences concerning steady-state performance. The reason is that the output data are typically correlated; consequently, the above statistical analysis, based on independent observations, is no longer applicable.

In order to cancel the effects of the time dependence and the initial distribution, it is common practice to discard the data that are collected during the nonstationary or transient part of the simulation. However, it is not always clear when the process will reach stationarity.

If the process is regenerative, then the regenerative method, discussed in Section 4.3.2.2, avoids this transience problem altogether.

From now on, we assume that $\{X_t\}$ is a stationary process. Suppose that we wish to estimate the steady-state expected value $\ell = \mathbb{E}[X_t]$, for example, the expected steady-state queue length, or the expected steady-state sojourn time of a customers in a queue. Then ℓ can be estimated as either

$$\widehat{\ell} = T^{-1} \sum_{t=1}^{T} X_t$$

or

$$\widehat{\ell} = T^{-1} \int_0^T X_t \, \mathrm{d}t \,,$$

respectively, depending on whether $\{X_t\}$ is a discrete-time or continuous-time process.

For concreteness, consider the discrete case. The variance of $\widehat{\ell}$ (see Problem 1.15) is given by

$$\mathrm{Var}(\widehat{\ell}) = \frac{1}{T^2} \left(\sum_{t=1}^{T} \mathrm{Var}(X_t) + 2 \sum_{s=1}^{T-1} \sum_{t=s+1}^{T} \mathrm{Cov}(X_s, X_t) \right) . \tag{4.16}$$

Since $\{X_t\}$ is stationary, we have $\mathrm{Cov}(X_s, X_t) = \mathbb{E}[X_s X_t] - \ell^2 = R(t-s)$, where R defines the *covariance function* of the stationary process. Note that $R(0) = \mathrm{Var}(X_t)$. As a consequence, we can write (4.16) as

$$T \, \mathrm{Var}(\widehat{\ell}) = R(0) + 2 \sum_{t=1}^{T-1} \left(1 - \frac{t}{T} \right) R(t) \,. \tag{4.17}$$

Similarly, if $\{X_t\}$ is a continuous-time process, the sum in (4.17) is replaced with the corresponding integral (from $t = 0$ to T), while all other data remain the same. In many applications $R(t)$ decreases rapidly with t, so that only the first few terms in the sum (4.17) are relevant. These covariances, say $R(0), R(1), \ldots, R(K)$, can be estimated via their (unbiased) sample averages:

$$\widehat{R}(k) = \frac{1}{T-k-1} \sum_{t=1}^{T-k} (X_t - \widehat{\ell})(X_{t+k} - \widehat{\ell}), \quad k = 0, 1, \ldots, K \,.$$

Thus, for large T the variance of $\widehat{\ell}$ can be estimated as $\tilde{S}^2/T$, where

$$\tilde{S}^2 = \widehat{R}(0) + 2 \sum_{t=1}^{K} \widehat{R}(t) \,.$$

To obtain confidence intervals, one again uses the central limit theorem, that is, the cdf of $\sqrt{T}(\widehat{\ell} - \ell)$ converges to the cdf of the normal distribution with expectation 0 and variance $\sigma^2 = \lim_{T \to \infty} T \, \mathrm{Var}(\widehat{\ell})$ — the so-called *asymptotic variance* of $\widehat{\ell}$. Using $\tilde{S}^2$ as an estimator for σ^2, we find that an approximate $(1-\alpha)100\%$ confidence interval for ℓ is given by

$$\left(\widehat{\ell} \pm z_{1-\alpha/2} \frac{\tilde{S}}{\sqrt{T}} \right) . \tag{4.18}$$

Below we consider two popular methods for estimating steady-state parameters: the *batch means* and *regenerative* methods.

4.3.2.1 The Batch Means Method The batch means method is most widely used by simulation practitioners to estimate steady-state parameters from a single simulation run, say of length M. The initial K observations, corresponding to the transient part of the run, are deleted, and the remaining $M - K$ observations are divided into N batches, each of length

$$T = \frac{M - K}{N} .$$

The deletion serves to eliminate or reduce the initial bias, so that the remaining observations $\{X_t, t > K\}$ are statistically more typical of the steady state.

Suppose we want to estimate the expected steady-state performance $\ell = \mathbb{E}[X_t]$, assuming that the process is stationary for $t > K$. Assume for simplicity that $\{X_t\}$ is a discrete-time process. Let X_{ti} denote the t-th observation from the i-th batch. The sample mean of the i-th batch of length T is given by

$$Y_i = \frac{1}{T} \sum_{t=1}^{T} X_{ti}, \quad i = 1, \ldots, N .$$

Therefore, the sample mean $\widehat{\ell}$ of ℓ is

$$\widehat{\ell} = \frac{1}{M - K} \sum_{t=K+1}^{M} X_t = \frac{1}{N} \sum_{i=1}^{N} Y_i . \tag{4.19}$$

The procedure is illustrated in Figure 4.4.

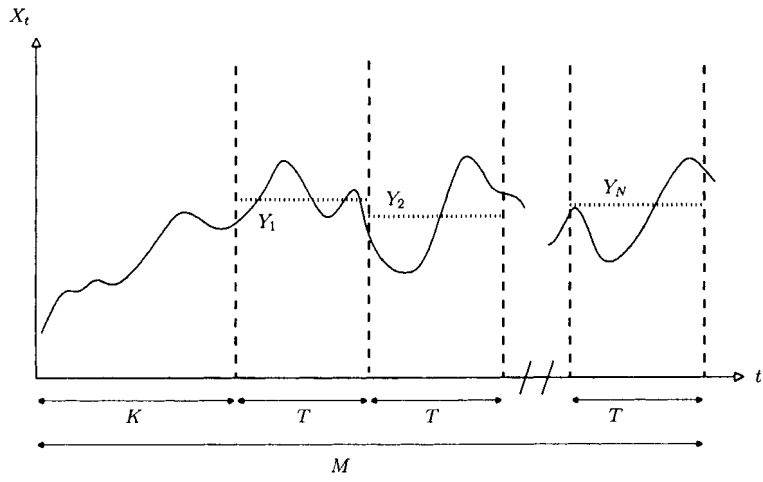


Figure 4.4 Illustration of the batch means procedure.

In order to ensure approximate independence between the batches, their size, T, should be large enough. In order for the central limit theorem to hold approximately, the number of batches, N, should typically be chosen in the range 20–30. In such a case, an approximate confidence interval for ℓ is given by (4.7), where S is the sample standard deviation of the $\{Y_i\}$. In the case where the batch means do exhibit some dependence, we can apply formula (4.18) as an alternative to (4.7).

Next, we shall discuss briefly how to choose K. In general, this is a very difficult task, since very few analytic results are available. The following queueing example provides some hints on how K should be increased as the traffic intensity in the queue increases.

Let $\{X_t, t \geq 0\}$ be the queue length process (not including the customer in service) in an $M/M/1$ queue, and assume that we start the simulation at time zero with an empty queue. It is shown in [1, 2] that in order to be within 1% of the steady-state mean, the length of the initial portion to be deleted, K, should be on the order of $8/(\mu(1-\varrho)^2)$, where $1/\mu$ is the expected service time. Thus, for $\varrho = 0.5$, 0.8, 0.9, and 0.95, K equals 32, 200, 800, and 3200 expected service times, respectively.

In general, one can use the following simple rule of thumb.

1. Define the following moving average A_k of length T:

$$A_k = \frac{1}{T} \sum_{t=k+1}^{T+k} X_t \, .$$

2. Calculate A_k for different values of k, say $k = 0, m, 2m, \ldots, rm, \ldots$, where m is fixed, say $m = 10$.

3. Find r such that $A_{rm} \approx A_{(r+1)m} \approx \cdots \approx A_{(r+s)m}$, while $A_{(r-s)m} \not\approx A_{(r-s+1)m} \not\approx \cdots \not\approx A_{rm}$, where $r \geq s$ and $s = 5$, for example.

4. Deliver $K = r\,m$.

The batch means algorithm is as follows:

Algorithm 4.3.2 (Batch Means Method)

1. *Make a single simulation run of length M and delete K observations corresponding to a finite-horizon simulation.*

2. *Divide the remaining $M - K$ observations into N batches, each of length*

$$T = \frac{M-K}{N} \, .$$

3. *Calculate the point estimator and the confidence interval for ℓ from (4.19) and* (4.7)*, respectively.*

■ EXAMPLE 4.4 $GI/G/1$ Queue

The $GI/G/1$ queueing model is a generalization of the $M/M/1$ model discussed in Examples 1.13 and 4.3. The only differences are that (1) the interarrival times each have a general cdf F and (2) the service times each have a general cdf G. Consider the process $\{Z_n, n = 1, 2, \ldots\}$ describing the number of people in a $GI/G/1$ queue as seen by the n-th arriving customer. Figure 4.5 gives a realization of the batch means procedure for estimating the steady-state queue length. In this example the first $K = 100$ observations are thrown away, leaving $N = 9$ batches, each of size $T = 100$. The batch means are indicated by thick lines.

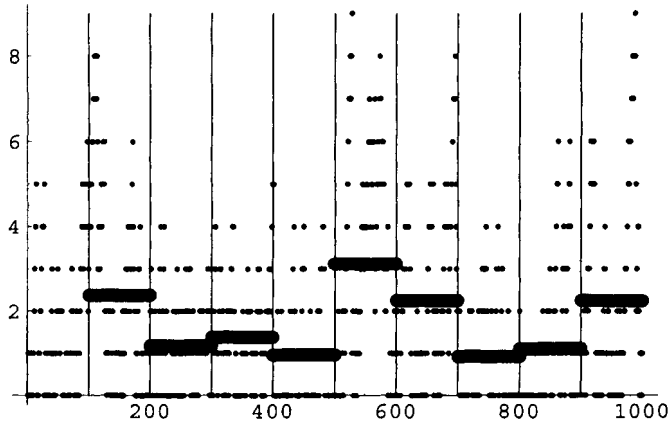


Figure 4.5 The batch means for the process $\{Z_n, n = 1, 2, \ldots\}$.

Remark 4.3.1 (Replication-Deletion Method) In the replication-deletion method N independent runs are carried out, rather than a single simulation run as in the batch means method. From each replication, one deletes K initial observations corresponding to the finite-horizon simulation and then calculates the point estimator and the confidence interval for ℓ via (4.19) and (4.7), respectively, exactly as in the batch means approach. Note that the confidence interval obtained with the replication-deletion method is unbiased, whereas the one obtained by the batch means method is slightly biased. However, the former requires deletion from *each* replication, as compared to *a single* deletion in the latter. For this reason, the former is not as popular as the latter. For more details on the replication-deletion method see [9].

4.3.2.2 The Regenerative Method A stochastic process $\{X_t\}$ is called *regenerative* if there exist random time points $T_0 < T_1 < T_2 < \ldots$ such that at each such time point the process restarts probabilistically. More precisely, the process $\{X_t\}$ can be split into iid replicas during intervals, called *cycles*, of lengths $\tau_i = T_i - T_{i-1}$, $i = 1, 2, \ldots$.

■ **EXAMPLE 4.5 Markov Chain**

The standard example of a regenerative process is a Markov chain. Assume that the chain starts from state i. Let $T_0 < T_1 < T_2 < \ldots$ denote the times that it visits state j. Note that at each random time T_n the Markov chain starts afresh, independently of the past. We say that the Markov process *regenerates* itself. For example, consider a two-state Markov chain with transition matrix

$$P = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix}. \tag{4.20}$$

Assume that all four transition probabilities p_{ij} are strictly positive and that, starting from state $i = 1$, we obtain the following sample trajectory:

$$(x_0, x_1, x_2, \ldots, x_{10}) = (1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 1).$$

It is readily seen that the transition probabilities corresponding to the above sample trajectory are

$$p_{12}, p_{22}, p_{22}, p_{21}, p_{12}, p_{21}, p_{11}, p_{12}, p_{22}, p_{21}.$$

Taking $j = 1$ as the regenerative state, the trajectory contains four cycles with the following transitions:

$$1 \to 2 \to 2 \to 2 \to 1; \quad 1 \to 2 \to 1; \quad 1 \to 1; \quad 1 \to 2 \to 2 \to 1,$$

and the corresponding cycle lengths are $\tau_1 = 4$, $\tau_2 = 2$, $\tau_3 = 1$, $\tau_4 = 3$.

■ **EXAMPLE 4.6** $GI/G/1$ **Queue (Continued)**

Another classic example of a regenerative process is the process $\{X_t, t \geqslant 0\}$ describing the number of customers in the $GI/G/1$ system, where the regeneration times $T_0 < T_1 < T_2 < \ldots$ correspond to customers arriving at an empty system (see also Example 4.4, where a related discrete-time process is considered). Observe that at each such time T_i the process starts afresh, independently of the past; in other words, the process regenerates itself. Figure 4.6 illustrates a typical sample path of the process $\{X_t, t \geqslant 0\}$. Note that here $T_0 = 0$, that is, at time 0 a customer arrives at an empty system.

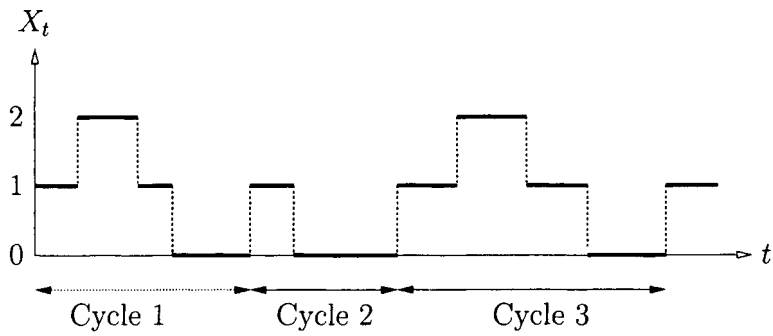


Figure 4.6 A sample path of the process $\{X_t, t \geqslant 0\}$, describing the number of customers in a $GI/G/1$ queue.

■ **EXAMPLE 4.7** (s, S) **Policy Inventory Model**

Consider a continuous-review, single-commodity inventory model supplying external demands and receiving stock from a production facility. When demand occurs, it is either filled or back-ordered (to be satisfied by delayed deliveries). At time t, the *net inventory* (on-hand inventory minus back orders) is N_t, and the *inventory position* (net inventory plus on-order inventory) is X_t. The control policy is an (s, S) policy that operates on the inventory position. Specifically, at any time t when a demand D is received that would reduce the inventory position to less than s (that is, $X_{t-} - D < s$, where X_{t-} denotes the inventory position just before t), an order of size $S - (X_{t-} - D)$ is placed, which brings the inventory position immediately back to S. Otherwise, no action is taken. The order arrives r time units after it is placed (r is called the *lead* time). Clearly, $X_t = N_t$ if $r = 0$. Both inventory processes are illustrated in Figure 4.7. The dots in the graph of the inventory position (below the s-line) represent what the inventory position would have been if no order was placed.

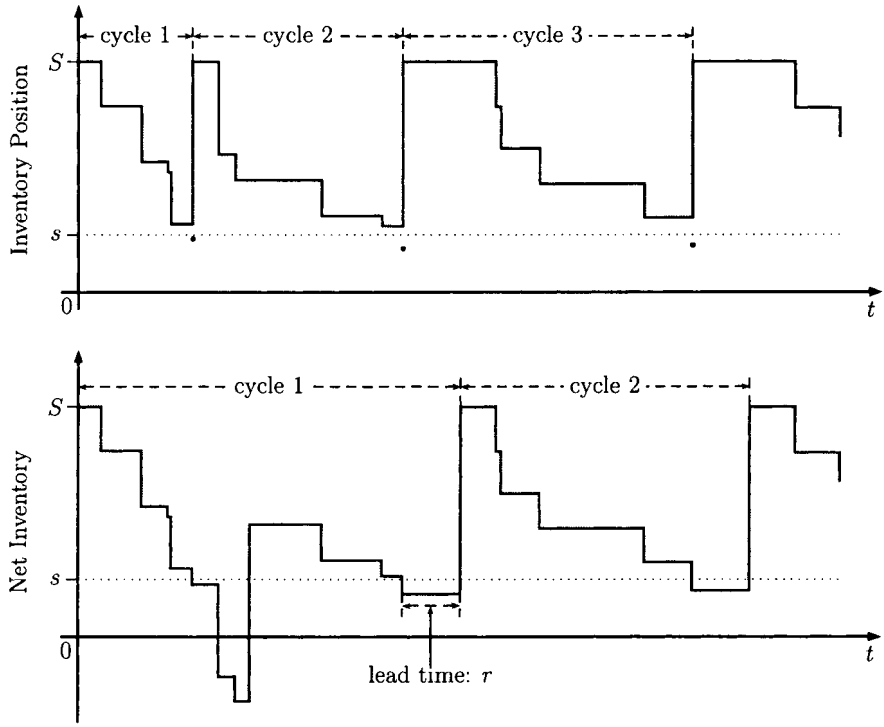


Figure 4.7 Sample paths for the two inventory processes.

Let D_i and A_i be the size of the i-th demand and the length of the i-th inter-demand time, respectively. We assume that both $\{D_i\}$ and $\{A_i\}$ are iid sequences, with common cdfs F and G, respectively. In addition, the sequences are assumed to be independent of each other. Under the back-order policy and the above assumptions, both the inventory position process $\{X_t\}$ and the net inventory process $\{N_t\}$ are regenerative. In particular, each process regenerates when it is raised to S. For example, each time an order is placed, the inventory position process regenerates. It is readily seen that the sample path of $\{X_t\}$ in Figure 4.7 contains three regenerative cycles, while the sample path of $\{N_t\}$ contains only two, which occur after the second and third lead times. Note that during these times no order has been placed.

The main strengths of the concept of regenerative processes are that the existence of limiting distributions is guaranteed under very mild conditions and the behavior of the limiting distribution depends only on the behavior of the process during a typical cycle.

Let $\{X_t\}$ be a regenerative process with regeneration times $T_0, T_1, T_2, \ldots$. Let $\tau_i = T_i - T_{i-1}, i = 1, 2, \ldots$ be the cycle lengths. Depending on whether $\{X_t\}$ is a discrete-time or continuous-time process, define, for some real-valued function H,

$$R_i = \sum_{t=T_{i-1}}^{T_i - 1} H(X_t) \tag{4.21}$$

or

$$R_i = \int_{T_{i-1}}^{T_i} H(X_t)\,\mathrm{d}t\,, \tag{4.22}$$

respectively, for $i = 1, 2, \ldots$. We assume for simplicity that $T_0 = 0$. We also assume that in the discrete case the cycle lengths are not always a multiple of some integer greater than 1. We can view R_i as the reward (or, alternatively, the cost) accrued during the i-th cycle. Let $\tau = T_1$ be the length of the first regeneration cycle and let $R = R_1$ be the first reward.

The following properties of regenerative processes will be needed later on; see, for example, [3].

(a) If $\{X_t\}$ is regenerative, then the process $\{H(X_t)\}$ is regenerative as well.

(b) If $\mathbb{E}[\tau] < \infty$, then, under mild conditions, the process $\{X_t\}$ has a limiting (or steady-state) distribution, in the sense that there exists a random variable X, such that

$$\lim_{t\to\infty} \mathbb{P}(X_t \leqslant x) = \mathbb{P}(X \leqslant x)\,.$$

In the discrete case, no extra condition is required. In the continuous case a sufficient condition is that the sample paths of the process are right-continuous and that the cycle length distribution is *non-lattice* — that is, the distribution does not concentrate all its probability mass at points $n\delta$, $n \in \mathbb{N}$, for some $\delta > 0$.

(c) If the conditions in (b) hold, the steady-state expected value, $\ell = \mathbb{E}[H(X)]$, is given by

$$\ell = \mathbb{E}[H(X)] = \frac{\mathbb{E}[R]}{\mathbb{E}[\tau]}. \tag{4.23}$$

(d) (R_i, τ_i), $i = 1, 2, \ldots$, is a sequence of iid random vectors.

Note that property (a) states that the behavior patterns of the system (or any measurable function thereof) during distinct cycles are statistically iid, while property (d) asserts that rewards and cycle lengths are jointly iid for distinct cycles. Formula (4.23) is fundamental to regenerative simulation. For typical non-Markovian queueing models, the quantity ℓ (the steady-state expected performance) is unknown and must be evaluated via regenerative simulation.

To obtain a point estimate of ℓ, one generates N regenerative cycles, calculates the iid sequence of two-dimensional random vectors (R_i, τ_i), $i = 1, \ldots, N$, and finally estimates ℓ by the *ratio* estimator

$$\widehat{\ell} = \frac{\widehat{R}}{\widehat{\tau}}\,, \tag{4.24}$$

where $\widehat{R} = N^{-1}\sum_{i=1}^{N} R_i$ and $\widehat{\tau} = N^{-1}\sum_{i=1}^{N} \tau_i$. Note that the estimator $\widehat{\ell}$ is biased, that is, $\mathbb{E}[\widehat{\ell}] \neq \ell$. However, $\widehat{\ell}$ is *strongly consistent*, that is, it converges to ℓ with probability 1 as $N \to \infty$. This follows directly from the fact that, by the law of large numbers, $\widehat{R}$ and $\widehat{\tau}$ converge with probability 1 to $\mathbb{E}[R]$ and $\mathbb{E}[\tau]$, respectively.

The *advantages* of the regenerative simulation method are:

(a) No deletion of transient data is necessary.

(b) It is asymptotically exact.

(b) It is easy to understand and implement.

The *disadvantages* of the regenerative simulation method are:

(a) For many practical cases, the output process, $\{X_t\}$, is either nonregenerative or its regeneration points are difficult to identify. Moreover, in complex systems (for example, large queueing networks), checking for the occurrence of regeneration points could be computationally expensive.

(b) The estimator $\widehat{\ell}$ is biased.

(c) The regenerative cycles may be very long.

Next, we shall establish a confidence interval for ℓ. Let $Z_i = R_i - \ell\tau_i$. It is readily seen that the Z_i are iid random variables, like the random vectors (R_i, τ_i). Letting $\widehat{R}$ and $\widehat{\tau}$ be defined as before, the central limit theorem ensures that

$$\frac{N^{1/2}\left(\widehat{R} - \ell\widehat{\tau}\right)}{\sigma} = \frac{N^{1/2}\left(\widehat{\ell} - \ell\right)}{\sigma/\widehat{\tau}}$$

converges in distribution to the standard normal distribution as $N \to \infty$, where

$$\sigma^2 = \mathrm{Var}(Z) = \mathrm{Var}(R) - 2\ell\,\mathrm{Cov}(R,\tau) + \ell^2\,\mathrm{Var}(\tau)\ . \tag{4.25}$$

Therefore, a $(1-\alpha)100\%$ confidence interval for $\ell = \mathbb{E}[R]/\mathbb{E}[\tau]$ is

$$\left(\widehat{\ell} \pm \frac{z_{1-\alpha/2}\,S}{\widehat{\tau}\,N^{1/2}}\right)\ , \tag{4.26}$$

where

$$S^2 = S_{11} - 2\,\widehat{\ell}\,S_{12} + \widehat{\ell}^{\,2}S_{22} \tag{4.27}$$

is the estimator of σ^2 based on replacing the unknown quantities in (4.25) with their unbiased estimators. That is,

$$S_{11} = \frac{1}{N-1}\sum_{i=1}^{N}(R_i - \widehat{R})^2,\quad S_{22} = \frac{1}{N-1}\sum_{i=1}^{N}(\tau_i - \widehat{\tau})^2$$

and

$$S_{12} = \frac{1}{N-1}\sum_{i=1}^{N}(R_i - \widehat{R})\,(\tau_i - \widehat{\tau})\ .$$

Note that (4.26) differs from the standard confidence interval, say (4.7), by having an additional term $\widehat{\tau}$.

The algorithm for estimating the $(1-\alpha)100\%$ confidence interval for ℓ is as follows:

Algorithm 4.3.3 (Regenerative Simulation Method)

1. *Simulate N regenerative cycles of the process $\{X_t\}$.*
2. *Compute the sequence $\{(R_i, \tau_i), i = 1, \ldots, N\}$.*
3. *Calculate the point estimator $\widehat{\ell}$ and the confidence interval of ℓ from (4.24) and (4.26), respectively.*

Note that if one uses two independent simulations of length N, one for estimating $\mathbb{E}[R]$ and the other for estimating $\mathbb{E}[\tau]$, then clearly $S^2 = S_{11} + \widehat{\ell}^{\,2} S_{22}$, since $\mathrm{Cov}(R, \tau) = 0$.

Remark 4.3.2 If the reward in each cycle is of the form (4.21) or (4.22), then $\ell = \mathbb{E}[H(X)]$ can be viewed as both the expected steady-state performance and the long-run average performance. This last interpretation is valid even if the reward in each cycle is not of the form (4.21)–(4.22) as long as the $\{(\tau_i, R_i)\}$ are iid. In that case,

$$\ell = \lim_{t \to \infty} \frac{\sum_{i=0}^{N_t - 1} R_i}{t} = \frac{\mathbb{E}[R]}{\mathbb{E}[\tau]}, \tag{4.28}$$

where N_t is the number of regenerations in $[0, t]$.

■ **EXAMPLE 4.8 Markov Chain: Example 4.5 (Continued)**

Consider again the two-state Markov chain with the transition matrix

$$P = \begin{pmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{pmatrix}.$$

Assume, as in Example 4.5, that starting from 1 we obtain the following sample trajectory: $(x_0, x_1, x_2, \ldots, x_{10}) = (1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 1)$, which has four cycles with lengths $\tau_1 = 4$, $\tau_2 = 2$, $\tau_3 = 1$, $\tau_4 = 3$ and corresponding transitions $(p_{12}, p_{22}, p_{22}, p_{21}), (p_{12}, p_{21}), (p_{11}), (p_{12}, p_{22}, p_{21})$. In addition, assume that each transition from i to j incurs a cost (or, alternatively, a reward) c_{ij} and that the related cost matrix is

$$C = (c_{ij}) = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}.$$

Note that the cost in each cycle is not of the form (4.21) (however, see Problem 4.14) but is given as

$$R_i = \sum_{t=T_{i-1}}^{T_i - 1} c_{X_t, X_{t+1}}, \quad i = 1, 2, \ldots.$$

We illustrate the estimation procedure for the long-run average cost ℓ. First, observe that $R_1 = 1 + 3 + 3 + 2 = 9$, $R_2 = 3$, $R_3 = 0$, and $R_4 = 6$. It follows that $\widehat{R} = 4.5$. Since $\widehat{\tau} = 2.5$, the point estimate of ℓ is $\widehat{\ell} = 1.80$. Moreover, $S_{11} = 15, S_{22} = 5/3, S_{12} = 5$, and $S^2 = 2.4$. This gives a 95% confidence interval for ℓ of $(1.20, 2.40)$.

■ **EXAMPLE 4.9 Example 4.6 (Continued)**

Consider the sample path in Figure 4.6 of the process $\{X_t, t \geqslant 0\}$ describing the number of customers in the $GI/G/1$ system. The corresponding sample path data are given in Table 4.2.

Table 4.2 Sample path data for the $GI/G/1$ queueing process.

$t \in$ interval	X_t	$t \in$ interval	X_t	$t \in$ interval	X_t
$[0.00, 0.80)$	1	$[3.91, 4.84)$	1	$[6.72, 7.92)$	1
$[0.80, 1.93)$	2	$[4.84, 6.72)$	0	$[7.92, 9.07)$	2
$[1.93, 2.56)$	1			$[9.07, 10.15)$	1
$[2.56, 3.91)$	0			$[10.15, 11.61)$	0
Cycle 1		**Cycle 2**		**Cycle 3**	

Notice that the figure and table reveal three complete cycles with the following pairs: $(R_1, \tau_1) = (3.69, 3.91)$, $(R_2, \tau_2) = (0.93, 2.81)$, and $(R_3, \tau_3) = (4.58, 4.89)$. The resultant statistics are (rounded) $\widehat{\ell} = 0.79$, $S_{11} = 3.62$, $S_{22} = 1.08$, $S_{12} = 1.92$, $S^2 = 1.26$ and the 95% confidence interval is (0.79 ± 0.32).

■ **EXAMPLE 4.10 Example 4.7 (Continued)**

Let $\{X_t, t \geqslant 0\}$ be the inventory position process described in Example 4.7. Table 4.3 presents the data corresponding to the sample path in Figure 4.7 for a case where $s = 10$, $S = 40$, and $r = 1$.

Table 4.3 The data for the inventory position process, $\{X_t\}$, with $s = 10$ and $S = 40$. The boxes indicate the regeneration times.

t	X_t	t	X_t	t	X_t
$\boxed{0.00}$	40.00	$\boxed{5.99}$	40.00	$\boxed{9.67}$	40.00
1.79	32.34	6.41	33.91	11.29	32.20
3.60	22.67	6.45	23.93	11.38	24.97
5.56	20.88	6.74	19.53	12.05	18.84
5.62	11.90	8.25	13.32	13.88	13.00
		9.31	10.51	$\boxed{14.71}$	40.00

Based on the data in Table 4.3, we illustrate the derivation of the point estimator and the 95% confidence interval for the steady-state quantity $\ell = \mathbb{P}(X < 30) = \mathbb{E}[I_{\{X<30\}}]$, that is, the probability that the inventory position is less than 30. Table 4.3 reveals three complete cycles with the following pairs: $(R_1, \tau_1) = (2.39, 5.99)$, $(R_2, \tau_2) = (3.22, 3.68)$, and $(R_3, \tau_3) = (3.33, 5.04)$, where $R_i = \int_{T_{i-1}}^{T_i} I_{\{X_t<30\}}\, dt$. The resulting statistics are (rounded) $\widehat{\ell} = 0.61$, $S_{11} = 0.26$, $S_{22} = 1.35$, $S_{12} = -0.44$, and $S^2 = 1.30$, which gives a 95% confidence interval (0.61 ± 0.26).

4.4 THE BOOTSTRAP METHOD

Suppose we estimate a number ℓ via some estimator $H = H(\mathbf{X})$, where $\mathbf{X} = (X_1, \ldots, X_n)$, and the $\{X_i\}$ form a random sample from some unknown distribution F. It

is assumed that H does not depend on the order of the $\{X_i\}$. To assess the quality (for example, accuracy) of the estimator H, one could draw independent replications $\mathbf{X}_1, \ldots, \mathbf{X}_N$ of $\mathbf{X}$ and find sample estimates for quantities such as the variance of the estimator

$$\text{Var}(H) = \mathbb{E}[H^2] - (\mathbb{E}[H])^2,$$

the *bias* of the estimator

$$\text{Bias} = \mathbb{E}[H] - \ell,$$

and the expected quadratic error, or *mean square error* (MSE)

$$\text{MSE} = \mathbb{E}\left[(H - \ell)^2\right].$$

However, it may be too time-consuming, or simply not feasible, to obtain such replications. An alternative is to *resample* the original data. Specifically, given an outcome $(x_1, \ldots, x_n)$ of $\mathbf{X}$, we draw a random sample $X_1^*, \ldots, X_n^*$ not from F but from an approximation to this distribution. The best estimate that we have about F on the grounds of $\{x_i\}$ is the *empirical distribution*, F_n, which assigns probability mass $1/n$ to each point $x_i, i = 1, \ldots, n$. In the one-dimensional case, the cdf of the empirical distribution is thus given by

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I_{\{x \leqslant x_i\}}.$$

Drawing from this distribution is trivial: for each j, draw $U \sim \text{U}[0,1]$, let $J = \lfloor U n \rfloor + 1$, and return $X_j^* = x_J$. Note that if the $\{x_i\}$ are all different, vector $\mathbf{X}^* = (X_1^*, \ldots, X_n^*)$ can take n^n different values.

The rationale behind the resampling idea is that the empirical distribution F_n is close to the actual distribution F and gets closer as n gets larger. Hence, any quantities depending on F, such as $\mathbb{E}_F[h(H)]$, where h is a function, can be approximated by $\mathbb{E}_{F_n}[h(H)]$. The latter is usually still difficult to evaluate, but it can be simply estimated via Monte Carlo simulation as

$$\frac{1}{B} \sum_{i=1}^{B} h(H_i^*),$$

where $H_1^*, \ldots, H_B^*$ are independent copies of $H^* = H(\mathbf{X}^*)$. This seemingly self-referent procedure is called *bootstrapping* — alluding to Baron von Münchhausen, who pulled himself out of a swamp by his own bootstraps. As an example, the bootstrap estimate of the expectation of H is

$$\widehat{\mathbb{E}[H]} = \overline{H}^* = \frac{1}{B} \sum_{i=1}^{B} H_i^*,$$

which is simply the sample mean of $\{H_i^*\}$. Similarly, the bootstrap estimate for $\text{Var}(H)$ is the sample variance

$$\widehat{\text{Var}(H)} = \frac{1}{B-1} \sum_{i=1}^{B} (H_i^* - \overline{H}^*)^2. \tag{4.29}$$

Perhaps of more interest are the bootstrap estimators for the bias and MSE, respectively $\overline{H}^* - H$ and

$$\frac{1}{B} \sum_{i=1}^{B} (H_i^* - H)^2.$$

Note that for these estimators the unknown quantity ℓ is replaced with the original estimator H. Confidence intervals can be constructed in the same fashion. We discuss two variants: the *normal* method and the *percentile* method. In the normal method, a $(1-\alpha)100\%$ confidence interval for ℓ is given by

$$(H \pm z_{1-\alpha/2} S^*) ,$$

where S^* is the bootstrap estimate of the standard deviation of H, that is, the square root of (4.29). In the percentile method, the upper and lower bounds of the $(1-\alpha)100\%$ confidence interval for ℓ are given by the $1-\alpha/2$ and $\alpha/2$ quantiles of H, which in turn are estimated via the corresponding sample quantiles of the bootstrap sample $\{H_i^*\}$.

PROBLEMS

4.1 We wish to estimate $\ell = \int_{-2}^{2} \mathrm{e}^{-x^2/2}\, dx = \int H(x) f(x)\, dx$ via Monte Carlo simulation using two different approaches: (a) defining $H(x) = 4\,\mathrm{e}^{-x^2/2}$ and f the pdf of the $\mathsf{U}[-2,2]$ distribution and (b) defining $H(x) = \sqrt{2\pi}\, I_{\{-2 \leqslant x \leqslant 2\}}$ and f the pdf of the $\mathsf{N}(0,1)$ distribution.

- **a)** For both cases, estimate ℓ via the estimator $\widehat{\ell}$ in (4.2). Use a sample size of $N = 100$.
- **b)** For both cases, estimate the relative error of $\widehat{\ell}$, using $N = 100$.
- **c)** Give a 95% confidence interval for ℓ for both cases, using $N = 100$.
- **d)** From b), assess how large N should be such that the relative width of the confidence interval is less than 0.001, and carry out the simulation with this N. Compare the result with the true value of ℓ.

4.2 Prove that the structure function of the bridge system in Figure 4.1 is given by (4.3).

4.3 Consider the bridge system in Figure 4.1. Suppose all link reliabilities are p. Show that the reliability of the system is $p^2(2 + 2p - 5p^2 + 2p^3)$.

4.4 Estimate the reliability of the bridge system in Figure 4.1 via (4.2) if the link reliabilities are $(p_1, \ldots, p_5) = (0.7, 0.6, 0.5, 0.4, 0.3)$. Choose a sample size such that the estimate has a relative error of about 0.01.

4.5 Consider the following sample performance:

$$H(\mathbf{X}) = \min\{X_1 + X_2,\ X_1 + X_4 + X_5,\ X_3 + X_4\}.$$

Assume that the random variables X_i, $i = 1, \ldots, 5$ are iid with common distribution

(a) $\mathsf{Gamma}(\lambda_i, \beta_i)$, where $\lambda_i = i$ and $\beta_i = i$.

(b) $\mathsf{Ber}(p_i)$, where $p_i = 1/2i$.

Run a computer simulation with $N = 1000$ replications, and find point estimates and 95% confidence intervals for $\ell = \mathbb{E}[H(\mathbf{X})]$.

4.6 Consider the precedence ordering of activities in Table 4.4. Suppose that durations of the activities (when actually started) are independent of each other, and all have exponential distributions with parameters 1.1, 2.3, 1.5, 2.9, 0.7, and 1.5, for activities $1, \ldots, 6$, respectively.

Table 4.4 Precedence ordering of activities.

Activity	1	2	3	4	5	6
Predecessor(s)	-	-	1	2,3	2,3	5

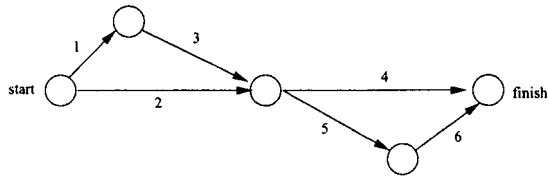


Figure 4.8 The PERT network corresponding to Table 4.4.

a) Verify that the corresponding PERT graph is given by Figure 4.8.

b) Identify the four possible paths from start to finish.

c) Estimate the expected length of the critical path in (4.5) with a relative error of less than 5%.

4.7 Let $\{X_t, t = 0, 1, 2, \ldots\}$ be a random walk on the positive integers; see Example 1.11. Suppose that $p = 0.55$ and $q = 0.45$. Let $X_0 = 0$. Let Y be the maximum position reached after 100 transitions. Estimate the probability that $Y \geqslant 15$ and give a 95% confidence interval for this probability based on 1000 replications of Y.

4.8 Consider the $M/M/1$ queue. Let X_t be the number of customers in the system at time $t \geqslant 0$. Run a computer simulation of the process $\{X_t, t \geqslant 0\}$ with $\lambda = 1$ and $\mu = 2$, starting with an empty system. Let X denote the steady-state number of people in the system. Find point estimates and confidence intervals for $\ell = \mathbb{E}[X]$, using the batch means and regenerative methods as follows:

a) For the batch means method run the system for a simulation time of 10,000, discard the observations in the interval $[0, 100]$, and use $N = 30$ batches.

b) For the regenerative method, run the system for the same amount of simulation time (10,000) and take as regeneration points the times where an arriving customer finds the system empty.

c) For both methods, find the requisite simulation time that ensures a relative width of the confidence interval not exceeding 5%.

4.9 Let Z_n be the number of customers in an $M/M/1$ queueing system, as seen by the n-th arriving customer, $n = 1, 2, \ldots$. Suppose that the service rate is $\mu = 1$ and the arrival rate is $\lambda = 0.6$. Let Z be the steady-state queue length (as seen by an arriving customer far away in the future). Note that $Z_n = X_{T_n-}$, with X_t as in Problem 4.8, and T_n is the arrival epoch of the n-th customer. Here, "T_n-" denotes the time just before T_n.

a) Verify that $\ell = \mathbb{E}[Z] = 1.5$.

b) Explain how to generate $\{Z_n, n = 1, 2, \ldots\}$ using a random walk on the positive integers, as in Problem 4.7.

c) Find the point estimate of ℓ and a 95% confidence interval for ℓ using the batch means method. Use a sample size of 10^4 customers and $N = 30$ batches, throwing away the first $K = 100$ observations.

d) Do the same as in c) using the regenerative method instead.

e) Assess the minimum length of the simulation run in order to obtain a 95% confidence interval with an absolute width w_a not exceeding 5%.

f) Repeat c), d), and e) with $\varrho = 0.8$ and discuss c), d), and e) as $\varrho \to 1$.

4.10 Table 4.5 displays a realization of a Markov chain, $\{X_t, t = 0, 1, 2, \ldots\}$, with state space $\{0, 1, 2, 3\}$ starting at 0. Let X be distributed according to the limiting distribution of this chain (assuming it has one).

Table 4.5 A realization of the Markov chain.

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X_t	0	3	0	1	2	1	0	2	0	1	0	1	0	2	0

Find the point estimator, $\widehat{\ell}$, and the 95% confidence interval for $\ell = \mathbb{E}[X]$ using the regenerative method.

4.11 Let W_n be the *waiting time* of the n-th customer in a $GI/G/1$ queue, that is, the total time the customer spends waiting in the queue (thus excluding the service time). The waiting time process $\{W_n, n = 1, 2, \ldots\}$ follows the following well-known *Lindley equation*:

$$W_{n+1} = \max\{W_n + S_n - A_{n+1}, 0\}, \quad n = 1, 2, \ldots, \tag{4.30}$$

where A_{n+1} is the interval between the n-th and $(n+1)$-st arrivals, S_n is the service time of the n-th customer, and $W_1 = 0$ (the first customer does not have to wait and is served immediately).

a) Explain why the Lindley equation holds.

b) Find the point estimate and the 95% confidence interval for the expected waiting time for the 4-th customer in an $M/M/1$ queue with $\varrho = 0.5$, $(\lambda = 1)$, starting with an empty system. Use $N = 5000$ replications.

c) Find point estimates and confidence intervals for the expected average waiting time for customers $21, \ldots, 70$ in the same system as in b). Use $N = 5000$ replications. Hint: the point estimate and confidence interval required are for the following parameter:

$$\ell = \mathbb{E}\left[\frac{1}{50}\sum_{n=21}^{70} W_n\right].$$

4.12 Run a computer simulation of 1000 regenerative cycles of the (s, S) policy inventory model (see Example 4.7), where demands arrive according to a Poisson process with rate 2 (that is, $A \sim \text{Exp}(2)$) and the size of each demand follows a Poisson distribution with mean 2 (that is, $D \sim \text{Poi}(2)$). Take $s = 1$, $S = 6$, lead time $r = 2$, and initial value $X_0 = 4$. Find point estimates and confidence intervals for the quantity $\ell = \mathbb{P}(2 \leqslant X \leqslant 4)$, where X is the steady-state inventory position.

4.13 Simulate the Markov chain $\{X_n\}$ in Example 4.8, using $p_{11} = 1/3$ and $p_{22} = 3/4$ for 1000 regeneration cycles. Obtain a confidence interval for the long-run average cost.

4.14 Consider Example 4.8 again, with $p_{11} = 1/3$ and $p_{22} = 3/4$. Define $Y_i = (X_i, X_{i+1})$ and $H(Y_i) = c_{X_i, X_{i+1}}$, $i = 0, 1, \ldots$. Show that $\{Y_i\}$ is a regenerative pro-

cess. Find the corresponding limiting/steady-state distribution and calculate $\ell = \mathbb{E}[H(Y)]$, where Y is distributed according to this limiting distribution. Check if ℓ is contained in the confidence interval obtained in Problem 4.13.

4.15 Consider the tandem queue of Section 3.3.1. Let X_t and Y_t denote the number of customers in the first and second queues at time t, including those who are possibly being served. Is $\{(X_t, Y_t), t \geqslant 0\}$ a regenerative process? If so, specify the regeneration times.

4.16 Consider the machine repair problem in Problem 3.5, with three machines and two repair facilities. Each repair facility can take only one failed machine. Suppose that the lifetimes are $\mathrm{Exp}(1/10)$ distributed and the repair times are $\mathrm{U}(0, 8)$ distributed. Let ℓ be the limiting probability that all machines are out of order.

a) Estimate ℓ via the regenerative estimator $\widehat{\ell}$ in (4.24) using 100 regeneration cycles. Compute the 95% confidence interval (4.27).

b) Estimate the bias and MSE of $\widehat{\ell}$ using the bootstrap method with a sample size of $B = 300$. (Hint: the original data are $\mathbf{X} = (X_1, \ldots, X_{100})$, where $X_i = (R_i, \tau_i)$, $i = 1, \ldots, 100$. Resample from these data using the empirical distribution.)

c) Compute 95% bootstrap confidence intervals for ℓ using the normal and percentile methods with $B = 1000$ bootstrap samples.

Further Reading

The regenerative method in a simulation context was introduced and developed by Crane and Iglehart [4, 5]. A more complete treatment of regenerative processes is given in [3]. Fishman [7] treats the statistical analysis of simulation data in great detail. Gross and Harris [8] is a classical reference on queueing systems. Efron and Tibshirani [6] gives the defining introduction to the bootstrap method.

REFERENCES

1. J. Abate and W. Whitt. Transient behavior of regulated Brownian motion, I: starting at the origin. *Advances in Applied Probability*, 19:560–598, 1987.
2. J. Abate and W. Whitt. Transient behavior of regulated Brownian motion, II: non-zero initial conditions. *Advances in Applied Probability*, 19:599–631, 1987.
3. S. Asmussen. *Applied Probability and Queues*. John Wiley & Sons, New York, 1987.
4. M. A. Crane and D. L. Iglehart. Simulating stable stochastic systems, I: general multiserver queues. *Journal of the ACM*, 21:103–113, 1974.
5. M. A. Crane and D. L. Iglehart. Simulating stable stochastic systems, II: Markov chains. *Journal of the ACM*, 21:114–123, 1974.
6. B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, 1994.
7. G. S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, New York, 1996.
8. D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 2nd edition, 1985.
9. A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, 3rd edition, 2000.

CHAPTER 5

CONTROLLING THE VARIANCE

5.1 INTRODUCTION

This chapter treats basic theoretical and practical aspects of *variance reduction* techniques. Variance reduction can be viewed as a means of utilizing known information about the model in order to obtain more accurate estimators of its performance. Generally, the more we know about the system, the more effective is the variance reduction. One way of gaining this information is through a pilot simulation run of the model. Results from this first-stage simulation can then be used to formulate variance reduction techniques that will subsequently improve the accuracy of the estimators in the second simulation stage. The main and most effective techniques for variance reduction are *importance sampling* and *conditional Monte Carlo*. Other well-known techniques that can provide moderate variance reduction include the use of common and antithetic variables, control variables, and stratification.

The rest of this chapter is organized as follows. We start, in Sections 5.2–5.5, with common and antithetic variables, control variables, conditional Monte Carlo, and stratified sampling. However, most of our attention, from Section 5.6 on, is focused on *importance sampling* and *likelihood ratio* techniques. Using importance sampling, one can often achieve substantial (sometimes dramatic) variance reduction, in particular when estimating rare-event probabilities. In Section 5.6 we present two alternative importance sampling-based techniques, called the *variance minimization* and *cross-entropy* methods. Section 5.7 discusses how importance sampling can be carried out sequentially/dynamically. Section 5.8 presents a simple, convenient, and unifying way of constructing efficient importance

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

sampling estimators: the so-called *transform likelihood ratio* (TLR) method. Finally, in Section 5.9 we present the *screening* method for variance reduction, which can also be seen as a dimension-reduction technique. The aim of this method is to identify (screen out) the most important (bottleneck) parameters of the simulated system to be used in an importance sampling estimation procedure.

5.2 COMMON AND ANTITHETIC RANDOM VARIABLES

To motivate the use of common and antithetic random variables in simulation, consider the following simple example. Let X and Y be random variables with known cdfs, F and G, respectively. Suppose we want to estimate $\ell = \mathbb{E}[X - Y]$ via simulation. The simplest unbiased estimator for ℓ is $X - Y$. Suppose we draw X and Y via the IT method:

$$\begin{aligned} X &= F^{-1}(U_1)\,, \quad U_1 \sim \mathsf{U}(0,1)\,, \\ Y &= G^{-1}(U_2)\,, \quad U_2 \sim \mathsf{U}(0,1)\,. \end{aligned} \tag{5.1}$$

The important point to notice is that X and Y (or U_1 and U_2) *need not be independent*. In fact, since

$$\mathrm{Var}(X - Y) = \mathrm{Var}(X) + \mathrm{Var}(Y) - 2\,\mathrm{Cov}(X, Y) \tag{5.2}$$

and since the marginal cdfs of X and Y have been prescribed, it follows that the variance of $X - Y$ is minimized by maximizing the covariance in (5.2). We say that *common random variables* are used in (5.1) if $U_2 = U_1$ and that *antithetic random variables* are used if $U_2 = 1 - U_1$. Since both F^{-1} and G^{-1} are nondecreasing functions, it is readily seen that using common random variables implies

$$\mathrm{Cov}\left(F^{-1}(U),\ G^{-1}(U)\right) \geqslant 0$$

for $U \sim \mathsf{U}(0,1)$. Consequently, variance reduction is achieved, in the sense that the estimator $F^{-1}(U) - G^{-1}(U)$ has a smaller variance than the *crude Monte Carlo* (CMC) estimator $X - Y$, where X and Y are independent, with cdfs F and G, respectively. In fact, it is well known (see, for example, [35]) that using common random variables maximizes the covariance between X and Y, so that $\mathrm{Var}(X - Y)$ is *minimized*. Similarly, $\mathrm{Var}(X + Y)$ is minimized when antithetic random variables are used.

Now consider minimal variance estimation of $\mathbb{E}[H_1(X) - H_2(Y)]$, where X and Y are unidimensional variables with known marginal cdfs, F and G, respectively, and H_1 and H_2 are real-valued monotone functions. Mathematically, the problem can be formulated as follows:

> Within the set of all two-dimensional joint cdfs of (X, Y), find a joint cdf, F^*, that minimizes $\mathrm{Var}(H_1(X) - H_2(Y))$, subject to X and Y having the prescribed cdfs F and G, respectively.

This problem has been solved by Gal, Rubinstein, and Ziv [11], who proved that if H_1 and H_2 are monotonic in the *same* direction, then the use of common random variables leads to optimal variance reduction, that is,

$$\min_{F^*} \mathrm{Var}(H_1(X) - H_2(Y)) = \mathrm{Var}\left(H_1[F^{-1}(U)] - H_2[G^{-1}(U)]\right)\,. \tag{5.3}$$

The proof of (5.3) uses the fact that if $H(u)$ is a monotonic function, then $H(F^{-1}(U))$ is monotonic as well, since $F^{-1}(u)$ is. By symmetry, if H_1 and H_2 are monotonic in *opposite*

directions, then the use of antithetic random variables (that is, $U_2 = 1 - U_1$) yields optimal variance reduction.

This result can be further generalized by considering minimal variance estimation of

$$\mathbb{E}[H_1(\mathbf{X}) - H_2(\mathbf{Y})] , \tag{5.4}$$

where $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_n)$ are random vectors with $X_i \sim F_i$ and $Y_i \sim G_i$, $i = 1, \ldots, n$, and the functions H_1 and H_2 are real-valued and monotone in each component of $\mathbf{X}$ and $\mathbf{Y}$. If the pairs $\{(X_i, Y_i)\}$ are independent and H_1 and H_2 are monotonic in the same direction (for each component), then the use of common random variables again leads to minimal variance. That is, we take $X_i = F_i^{-1}(U_i)$ and $Y_i = G_i^{-1}(U_i)$, $i = 1, \ldots, n$, where $U_1, \ldots, U_n$ are independent U(0, 1)-distributed random variables, or, symbolically,

$$\mathbf{X} = F^{-1}(\mathbf{U}), \quad \mathbf{Y} = G^{-1}(\mathbf{U}) . \tag{5.5}$$

Similarly, if H_1 and H_2 are monotonic in opposite directions, then using antithetic random variables is optimal. Finally, if H_1 and H_2 are monotonically increasing with respect to some components and monotonically decreasing with respect to others, then minimal variance is obtained by using the appropriate combination of common and antithetic random variables.

We now describe one of the main applications of antithetic random variables. Suppose one wants to estimate

$$\ell = \mathbb{E}[H(\mathbf{X})] ,$$

where $\mathbf{X} \sim F$ is a random vector with independent components and the sample performance function, $H(\mathbf{x})$, is monotonic in each component of $\mathbf{x}$. An example of such a function is given below.

■ EXAMPLE 5.1 Stochastic Shortest Path

Consider the undirected graph in Figure 5.1, depicting a so-called *bridge network*.

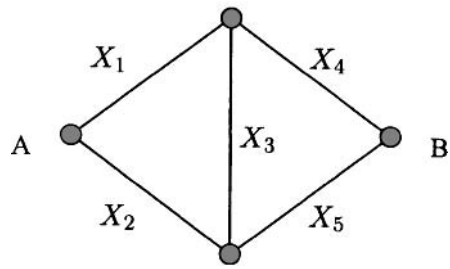


Figure 5.1 Determine the shortest path from A to B in a bridge network.

Suppose we wish to estimate the expected length ℓ of the shortest path between nodes (vertices) A and B, where the lengths of the links (edges) are random variables $X_1, \ldots, X_5$. We have $\ell = \mathbb{E}[H(\mathbf{X})]$, where

$$H(\mathbf{X}) = \min\{X_1 + X_4,\ X_1 + X_3 + X_5,\ X_2 + X_3 + X_4,\ X_2 + X_5\} . \tag{5.6}$$

Note that $H(\mathbf{x})$ is nondecreasing in each component of the vector $\mathbf{x}$.

Similarly, the length of the shortest path $H(\mathbf{X})$ in an arbitrary network with random edge lengths $\{X_i\}$ can be written as

$$H(\mathbf{X}) = \min_{j=1,\ldots,p} \sum_{i \in \mathscr{P}_j} X_i \,, \tag{5.7}$$

where $\mathscr{P}_j$ is the j-th complete path from the source to the sink of the network and p is the number of complete paths in the network. The sample performance is nondecreasing in each of the components.

An unbiased estimator of $\ell = \mathbb{E}[H(\mathbf{X})]$ is the CMC estimator, given by

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k) \,, \tag{5.8}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is an iid sample from the (multidimensional) cdf F. An alternative unbiased estimator of ℓ, for N even, is

$$\widehat{\ell}^{(a)} = \frac{1}{N} \sum_{k=1}^{N/2} \left\{ H(\mathbf{X}_k) + H(\mathbf{X}_k^{(a)}) \right\} , \tag{5.9}$$

where $\mathbf{X}_k = F^{-1}(\mathbf{U}_k)$ and $\mathbf{X}_k^{(a)} = F^{-1}(\mathbf{1} - \mathbf{U}_k)$, using notation similar to (5.5). The estimator $\widehat{\ell}^{(a)}$ is called the *antithetic estimator* of ℓ. Since $H(\mathbf{X}) + H(\mathbf{X}^{(a)})$ is a particular case of $H_1(\mathbf{X}) - H_2(\mathbf{Y})$ in (5.4) (with $H_2(\mathbf{Y})$ replaced by $-H(\mathbf{X}^{(a)})$), one immediately obtains that $\mathrm{Var}(\widehat{\ell}^{(a)}) \leqslant \mathrm{Var}(\widehat{\ell})$. That is, the antithetic estimator, $\widehat{\ell}^{(a)}$, is more accurate than the CMC estimator, $\widehat{\ell}$.

To compare the efficiencies of $\widehat{\ell}$ and $\widehat{\ell}^{(a)}$, one can consider their *relative time variance*,

$$\varepsilon = \frac{T^{(a)} \, \mathrm{Var}(\widehat{\ell}^{(a)})}{T \, \mathrm{Var}(\widehat{\ell})} \,, \tag{5.10}$$

where $T^{(a)}$ and T are the CPU times required to calculate the estimators $\widehat{\ell}^{(a)}$ and $\widehat{\ell}$, respectively. Note that

$$\begin{aligned} \mathrm{Var}(\widehat{\ell}^{(a)}) &= \frac{N/2}{N^2} \left(\mathrm{Var}(H(\mathbf{X})) + \mathrm{Var}(H(\mathbf{X}^{(a)})) + 2\,\mathrm{Cov}[H(\mathbf{X}), H(\mathbf{X}^{(a)})] \right) \\ &= \mathrm{Var}(\widehat{\ell}) + \mathrm{Cov}(H(\mathbf{X}), H(\mathbf{X}^{(a)}))/N \,. \end{aligned}$$

Also, $T^{(a)} \leqslant T$, since the antithetic estimator, $\widehat{\ell}^{(a)}$, needs only *half* as many random numbers as its CMC counterpart, $\widehat{\ell}$. Neglecting this time advantage, the efficiency measure (5.10) reduces to

$$\varepsilon = \frac{\mathrm{Var}(\widehat{\ell}^{(a)})}{\mathrm{Var}(\widehat{\ell})} = 1 + \frac{\mathrm{Cov}[H(\mathbf{X}), H(\mathbf{X}^{(a)})]}{\mathrm{Var}(H(\mathbf{X}))} \,, \tag{5.11}$$

where the covariance is negative and can be estimated via the corresponding sample covariance.

The use of common/antithetic random variables for the case of dependent components of $\mathbf{X}$ and $\mathbf{Y}$ for strictly monotonic functions, H_1 and H_2, is presented in Rubinstein, Samorodnitsky, and Shaked [33].

■ **EXAMPLE 5.2 Stochastic Shortest Path (Continued)**

We estimate the expected length of the shortest path for the bridge network in Example 5.1 for the case where each link has an exponential weight with parameter 1. Taking a sample size of $N = 10{,}000$, the CMC estimate is $\widehat{\ell} = 1.159$ with an estimated variance of $5.6 \cdot 10^{-5}$, whereas the antithetic estimate is $\widehat{\ell} = 1.164$ with an estimated variance of $2.8 \cdot 10^{-5}$. Therefore, the efficiency ε of the estimator $\widehat{\ell}^{(a)}$ relative to the CMC estimator $\widehat{\ell}$ is about 2.0.

■ **EXAMPLE 5.3 Lindley's Equation**

Consider Lindley's equation for the waiting time of the $(n+1)$-st customer in a $GI/G/1$ queue :

$$W_{n+1} = \max\{W_n + U_n, 0\}, \quad W_1 = 0 \, .$$

See also (4.30). Here $U_n = S_n - A_{n+1}$, where S_n is the service time of the n-th customer, and A_{n+1} is the interarrival time between the n-th and $(n+1)$-st customer. Since W_n is a monotonic function of each component $A_2, \ldots, A_n$ and $S_1, \ldots, S_{n-1}$, one can obtain variance reduction by using antithetic random variables.

5.3 CONTROL VARIABLES

The *control variables* method is one of the most widely used variance reduction techniques. Consider first the one-dimensional case. Let X be an unbiased estimator of μ, to be obtained from a simulation run. A random variable C is called a *control variable* for X if it is correlated with X and its expectation, r, is known. The control variable C is used to construct an unbiased estimator of μ with a variance smaller than that of X. This estimator,

$$X_\alpha = X - \alpha(C - r) \, , \tag{5.12}$$

where α is a scalar parameter, is called the *linear control variable.* The variance of X_α is given by

$$\mathrm{Var}(X_\alpha) = \mathrm{Var}(X) - 2\alpha \, \mathrm{Cov}(X, C) + \alpha^2 \, \mathrm{Var}(C)$$

(see, for example, Problem 1.15). Consequently, the value α^* that minimizes $\mathrm{Var}(X_\alpha)$ is

$$\alpha^* = \frac{\mathrm{Cov}(X, C)}{\mathrm{Var}(C)} \, . \tag{5.13}$$

Typically, α^* needs to be estimated from the corresponding sample covariance and variance. Using α^*, the minimal variance is

$$\mathrm{Var}(X_{\alpha^*}) = (1 - \varrho_{XC}^2)\mathrm{Var}(X) \, , \tag{5.14}$$

where ϱ_{XC} denotes the correlation coefficient of X and C. Notice that the larger $|\varrho_{XC}|$ is, the greater is the variance reduction.

Formulas (5.12)–(5.14) can be easily extended to the case of multiple control variables. Indeed, let $\mathbf{C} = (C_1, \ldots, C_m)^T$ be a (column) vector of m control variables with known mean vector $\mathbf{r} = \mathbb{E}[\mathbf{C}] = (r_1, \ldots, r_m)^T$, where $r_i = \mathbb{E}[C_i]$. Then the vector version of (5.12) can be written as

$$X_\alpha = X - \boldsymbol{\alpha}^T(\mathbf{C} - \mathbf{r}) \, , \tag{5.15}$$

where $\boldsymbol{\alpha}$ is an m-dimensional vector of parameters. It is not difficult to see that the value $\boldsymbol{\alpha}^*$ that minimizes $\mathrm{Var}(X_{\boldsymbol{\alpha}})$ is given by

$$\boldsymbol{\alpha}^* = \Sigma_C^{-1} \boldsymbol{\sigma}_{XC} , \tag{5.16}$$

where Σ_C denotes the $m \times m$ covariance matrix of $\mathbf{C}$ and $\boldsymbol{\sigma}_{XC}$ denotes the $m \times 1$ vector whose i-th component is the covariance of X and C_i, $i = 1, \ldots, m$. The corresponding minimal variance evaluates to

$$\mathrm{Var}(X_{\boldsymbol{\alpha}^*}) = (1 - R_{XC}^2)\mathrm{Var}(X) , \tag{5.17}$$

where

$$R_{XC}^2 = (\boldsymbol{\sigma}_{XC})^T \Sigma_C^{-1} \boldsymbol{\sigma}_{XC} / \mathrm{Var}(X)$$

is the square of the so-called *multiple correlation coefficient* of X and $\mathbf{C}$. Again, the larger $|R_{XC}|$ is, the greater is the variance reduction. The case where $\mathbf{X}$ is a vector with dependent components and the vector $\boldsymbol{\alpha}$ is replaced by a corresponding matrix is treated in Rubinstein and Marcus [30].

The following examples illustrate various applications of the control variables method.

■ **EXAMPLE 5.4 Stochastic Shortest Path (Continued)**

Consider again the stochastic shortest path estimation problem for the bridge network in Example 5.1. As control variables we can use, for example, the lengths of the paths $\mathscr{P}_j$, $j = 1, \ldots, 4$, that is, any (or all) of

$$\begin{aligned} C_1 &= X_1 + X_4 \\ C_2 &= X_1 + X_3 + X_5 \\ C_3 &= X_2 + X_3 + X_4 \\ C_4 &= X_2 + X_5 . \end{aligned}$$

The expectations of the $\{C_i\}$ are easy to calculate, and each C_i is positively correlated with the length of the shortest path $H(\mathbf{X}) = \min\{C_1, \ldots, C_4\}$.

■ **EXAMPLE 5.5 Lindley's Equation (Continued)**

Consider Lindley's equation for the waiting time process $\{W_n, n = 1, 2, \ldots\}$ in the $GI/G/1$ queue; see Example 5.3. As a control variable for W_n we can take C_n, defined by the recurrence relation

$$C_{n+1} = C_n + U_n, \qquad C_1 = 0 ,$$

where $U_n = S_n - A_{n+1}$, as in the waiting time process. Obviously, C_n and W_n are highly correlated. Moreover, the expectation $r_n = \mathbb{E}[C_n]$ is known. It is $r_n = (n-1)\,(\mathbb{E}[S] - \mathbb{E}[A])$, where $\mathbb{E}[S]$ and $\mathbb{E}[A]$ are the expected service and interarrival times, respectively. The corresponding linear control process is

$$Y_n = W_n - \alpha(C_n - r_n) .$$

■ **EXAMPLE 5.6 Queueing Networks**

Consider the estimation of the expected steady-state performance $\ell = \mathbb{E}[X]$ in a queueing network. For example, suppose that X is the steady-state number of customers in the system. As a linear control random process, one may take

$$Y_t = X_t - \alpha(C_t - r_t) ,$$

where X_t is the number of customers in the original system, and C_t is the number of customers in an auxiliary *Markovian* network for which the steady-state distribution is known. The latter network must be synchronized in time with the original network.

In order to produce high correlations between the two processes, $\{X_t\}$ and $\{C_t\}$, it is desirable that both networks have similar topologies and similar loads. In addition, they must use a common stream of random numbers for generating the input variables. Expressions for the expected steady-state performance $r = \mathbb{E}[C]$, such as the expected number in the system in a Markovian network, may be found in [14].

5.4 CONDITIONAL MONTE CARLO

Let

$$\ell = \mathbb{E}[H(\mathbf{X})] = \int H(\mathbf{x}) f(\mathbf{x}) \, d\mathbf{x} \tag{5.18}$$

be some expected performance measure of a computer simulation model, where $\mathbf{X}$ is the input random variable (vector) with a pdf $f(\mathbf{x})$ and $H(\mathbf{X})$ is the sample performance measure (output random variable). Suppose that there is a random variable (or vector), $\mathbf{Y} \sim g(\mathbf{y})$, such that the conditional expectation $\mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y} = \mathbf{y}]$ can be computed analytically. Since, by (1.11),

$$\ell = \mathbb{E}[H(\mathbf{X})] = \mathbb{E}[\, \mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y}] \,] , \tag{5.19}$$

it follows that $\mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y}]$ is an unbiased estimator of ℓ. Furthermore, it is readily seen that

$$\mathrm{Var}(\mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y}]) \leqslant \mathrm{Var}(H(\mathbf{X})) , \tag{5.20}$$

so that using the random variable $\mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y}]$, instead of $H(\mathbf{X})$, leads to variance reduction. Thus, conditioning *always* leads to variance reduction. To see (5.20), one uses the property (see Problem 5.6) that for any pair of random variables (U, V),

$$\mathrm{Var}(U) = \mathbb{E}[\, \mathrm{Var}(U \mid V)] + \mathrm{Var}(\, \mathbb{E}[U \mid V] \,) . \tag{5.21}$$

Since both terms on the right-hand side are nonnegative, (5.20) immediately follows. The conditional Monte Carlo idea is sometimes referred to as *Rao-Blackwellization*. The conditional Monte Carlo algorithm is given next.

Algorithm 5.4.1 (Conditional Monte Carlo)

1. *Generate a sample* $\mathbf{Y}_1, \ldots \mathbf{Y}_N$ *from* $g(\mathbf{y})$.
2. *Calculate* $\mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y}_k]$, $k = 1, \ldots, N$ *analytically.*
3. *Estimate* $\ell = \mathbb{E}[H(\mathbf{X})]$ *by*

$$\widehat{\ell}_c = \frac{1}{N} \sum^{N} \mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y}_k] . \tag{5.22}$$

Algorithm 5.4.1 requires that a random variable $\mathbf{Y}$ be found, such that $\mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y} = \mathbf{y}]$ is known analytically for all $\mathbf{y}$. Moreover, for Algorithm 5.4.1 to be of practical use, the following conditions must be met:

(a) $\mathbf{Y}$ should be easy to generate.

(b) $\mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y} = \mathbf{y}]$ should be readily computable for all values of $\mathbf{y}$.

(c) $\mathbb{E}[\mathrm{Var}(H(\mathbf{X}) \mid \mathbf{Y})]$ should be large relative to $\mathrm{Var}(\mathbb{E}[H(\mathbf{X}) \mid \mathbf{Y}])$.

■ **EXAMPLE 5.7 Random Sums**

Consider the estimation of

$$\ell = \mathbb{P}(S_R \leqslant x) = \mathbb{E}[I_{\{S_R \leqslant x\}}] ,$$

where

$$S_R = \sum_{i=1}^{R} X_i ,$$

R is a random variable with a given distribution and the $\{X_i\}$ are iid with $X_i \sim F$ and independent of R. Let F^r be the cdf of the random variable S_r for fixed $R = r$. Noting that

$$F^r(x) = \mathbb{P}\left(\sum_{i=1}^{r} X_i \leqslant x\right) = F\left(x - \sum_{i=2}^{r} X_i\right) ,$$

we obtain

$$\ell = \mathbb{E}\left[\mathbb{E}\left[I_{\{S_R \leqslant x\}} \,\middle|\, \sum_{i=2}^{R} X_i\right]\right] = \mathbb{E}\left[F\left(x - \sum_{i=2}^{R} X_i\right)\right] .$$

As an estimator of ℓ based on conditioning, we can take

$$\widehat{\ell}_c = \frac{1}{N} \sum_{k=1}^{N} F\left(x - \sum_{i=2}^{R_k} X_{ki}\right) . \tag{5.23}$$

5.4.1 Variance Reduction for Reliability Models

Next, we present two variance reduction techniques for reliability models based on conditioning. As in Example 4.1 on page 98, we are given an unreliable system of n components, each of which can be either functioning or failed, with a structure function H that determines the state of the system (working or failed) as a function of the states of the components. The component states $X_1, \ldots, X_n$ are assumed to be independent, with reliabilities $\{p_i\}$ and unreliabilities $\{q_i\}$, where $q_i = 1 - p_i$. The probability of system failure — the unreliability of the system — is thus $\overline{r} = \mathbb{P}(H(\mathbf{X}) = 0)$. In typical applications the unreliability is very small and is difficult to estimate via CMC.

5.4.1.1 Permutation Monte Carlo Permutation Monte Carlo is a conditional Monte Carlo technique for network reliability estimation; see Elperin et al. [9]. Here the components are unreliable links in a network, such as in Example 4.1. The system state $H(\mathbf{X})$ is the indicator of the event that certain preselected nodes are connected by functioning links. Suppose we wish to estimate the system unreliability $\bar{r} = \mathbb{P}(H(\mathbf{X}) = 0)$.

To apply the conditional Monte Carlo idea, we view the static network as a snapshot of a *dynamic* network at time $t = 1$. In this dynamic system, the links are repaired independently of each other with an exponential repair time with rate $\mu_e = -\ln(q_e), e = 1, \ldots, n$. At time $t = 0$ all links are failed. The state of the links at time t is given by the vector $\mathbf{X}_t$. Note that $\{\mathbf{X}_t, t \geqslant 0\}$ is a Markov jump process with state space $\{0,1\}^n$. Since the probability of each link e being operational at time $t = 1$ is p_e, the reliability of the dynamic network at time $t = 1$ is exactly the same as the reliability of the original network.

Let Π denote the *order* in which the links become operational, and let $S_0, S_0 + S_1, \ldots, S_0 + \cdots + S_{n-1}$ be the times at which those links are constructed. Π is a random variable that takes values in the space of permutations of the set of links $\mathcal{E} = \{1, \ldots, n\}$ — hence the name *permutation Monte Carlo*. For any permutation $\pi = (e_1, e_2, \ldots, e_n)$ define $\mathcal{E}_0 = \mathcal{E}$, and $\mathcal{E}_i = \mathcal{E}_{i-1} \setminus \{e_i\}$, $1 \leqslant i \leqslant n-1$. Thus, $\mathcal{E}_i$ corresponds to the set of links that are still failed after i links have been repaired. Let $b = b(\pi)$ be the number of repairs required (in the order defined by π) to bring the network up. This is called the *critical number* for π.

From the theory of Markov jump processes (see Section 1.12.5) it follows that

$$\mathbb{P}(\Pi = \pi) = \prod_{i=1}^{n} \frac{\mu_{e_i}}{\lambda_{i-1}}, \tag{5.24}$$

where $\lambda_i = \sum_{e \in \mathcal{E}_i} \mu_e$. More importantly, conditional on Π the sojourn times $S_0, \ldots, S_{n-1}$ are independent and each S_i is exponentially distributed with parameter $\lambda_i, i = 0, \ldots, n-1$. By conditioning on Π we have

$$\bar{r} = \sum_{\pi} \mathbb{P}[H(\mathbf{X}_1) = 0 \,|\, \Pi = \pi]\, \mathbb{P}[\Pi = \pi] = \mathbb{E}[g(\Pi)], \tag{5.25}$$

with

$$g(\pi) = \mathbb{P}[H(\mathbf{X}_1) = 0 \,|\, \Pi = \pi]. \tag{5.26}$$

From the definitions of S_i and b we see that $g(\pi)$ is equal to the probability that the sum of b independent exponential random variables with rates $\lambda_i, i = 0, 1, \ldots, b-1$ exceeds 1. This can be computed exactly, for example by using convolutions. Specifically, we have

$$g(\pi) = 1 - F_0 \star \cdots \star F_{b-1}(1),$$

where F_i is the cdf of the $\mathrm{Exp}(\lambda_i)$ distribution, and $\star$ means convolution; that is,

$$F \star G(t) = \int_0^t F(t-x)\, dG(x).$$

Alternatively, it can be shown (see, for example [25]) that

$$g(\pi) = (1, 0, \ldots, 0)\, \mathrm{e}^{A}\, (1, \ldots, 1)^T. \tag{5.27}$$

where A is the matrix with diagonal elements $-\lambda_0, \ldots, -\lambda_{b-1}$ and upper-diagonal elements $\lambda_0, \ldots, \lambda_{b-2}$ and 0 elsewhere. Here e^A is the *matrix exponential* $\sum_{k=0}^{\infty} A^k/k!$.

Let $\Pi_1, \ldots, \Pi_N$ be iid random permutations, each distributed according to Π; then

$$\widehat{\bar{r}} = \frac{1}{N} \sum_{k=1}^{N} g(\Pi_k) \tag{5.28}$$

is an unbiased estimator for $\bar{r}$. This leads to the following algorithm for estimating the unreliability $\bar{r}$.

Algorithm 5.4.2 (Permutation Monte Carlo)

1. *Draw a random permutation* Π *according to* (5.24). *A simple way, similar to Algorithm 2.8.1, is to draw* $Y_e \sim \mathsf{Exp}(\mu_e)$, $e = 1, \ldots, n$ *independently and return* Π *as the indices of the (increasing) ordered values.*
2. *Determine the critical number* b *and the rates* $\lambda_i, i = 1, \ldots, b-1$.
3. *Evaluate the conditional probability* $g(\Pi)$ *exactly, for example, via* (5.27).
4. *Repeat Steps 1–3 independently* N *times and deliver* (5.28) *as the estimator for* $\bar{r}$.

5.4.1.2 Conditioning Using Minimal Cuts The second method to estimate unreliability efficiently, developed by Ross [27], employs the concept of a minimal cut. A state vector $\mathbf{x}$ is called a *cut vector* if $H(\mathbf{x}) = 0$. If in addition $H(\mathbf{y}) = 1$ for all $\mathbf{y} > \mathbf{x}$, then $\mathbf{x}$ is called the *minimal cut vector*. Note that $\mathbf{y} > \mathbf{x}$ means that $y_i \geq x_i$, $i = 1, \ldots, n$ with $y_i > x_i$ for some i. If $\mathbf{x}$ is a minimal cut vector, the set $C = \{i : x_i = 0\}$ is called a *minimal cut set*. That is, a minimal cut set is a minimal set of components whose *failure* ensures the failure of the system. If $C_1, \ldots, C_m$ denote all the minimal cut sets, the system is functioning if and only if at least one component in each of the cut sets is functioning. It follows that $H(\mathbf{x})$ can be written as

$$H(\mathbf{x}) = \prod_{j=1}^{m} \max_{i \in C_j} x_i = \prod_{j=1}^{m} \left(1 - \prod_{i \in C_j} (1 - x_i) \right). \tag{5.29}$$

To proceed, we need the following proposition, which is adapted from [27].

Proposition 5.4.1 Let $Y_1, \ldots, Y_m$ be Bernoulli random variables (possibly dependent) with success parameters $a_1, \ldots, a_m$. Define $S = \sum_{j=1}^{m} Y_j$ and let $a = \mathbb{E}[S] = \sum_{j=1}^{m} a_j$. Let J be a discrete uniform random variable on $\{1, \ldots, m\}$ independent of $Y_1, \ldots, Y_m$. Finally, let R be any random variable that is independent of J. Then

$$\mathbb{P}(J = j \,|\, Y_J = 1) = \frac{a_j}{a}, \quad j = 1, \ldots, m \tag{5.30}$$

and

$$\mathbb{E}[SR] = \mathbb{E}[S]\, \mathbb{E}[R \,|\, Y_J = 1]\,. \tag{5.31}$$

Proof: To derive formula (5.30) write, using Bayes' formula,

$$\mathbb{P}(J = j \,|\, Y_J = 1) = \frac{\mathbb{P}(Y_J = 1 \,|\, J = j)\, \mathbb{P}(J = j)}{\sum_{i=1}^{m} \mathbb{P}(Y_J = 1 \,|\, J = i)\, \mathbb{P}(J = i)}\,.$$

Taking into account that $\mathbb{P}(Y_J = 1 \mid J = j) = \mathbb{P}(Y_j = 1 \mid J = j) = \mathbb{P}(Y_j = 1) = a_j$, the result follows. To prove (5.31), write

$$\mathbb{E}[SR] = \sum_{j=1}^{m} \mathbb{E}[RY_j] = \sum_{j=1}^{m} \mathbb{E}[R \mid Y_j = 1]\,\mathbb{P}(Y_j = 1)$$
$$= a \sum_{j=1}^{m} \mathbb{E}[R \mid Y_j = 1]\,\frac{a_j}{a}\,.$$

Since $a = \mathbb{E}[S]$ and, by (5.30), $\{a_j/a\}$ is the conditional distribution of J given $Y_J = 1$, (5.31) follows. □

We shall apply Proposition 5.4.1 to the estimation of the unreliability $\bar{r} = \mathbb{P}(H(\mathbf{X}) = 0)$. Let $Y_j = \prod_{i \in C_j}(1 - X_i)$, $j = 1, \ldots, m$, where, as before, $\{C_j\}$ denotes the collection of minimal cut sets. Thus, Y_j is the indicator of the event that all components in C_j are *failed*. Note that $Y_j \sim \mathsf{Ber}(a_j)$, with

$$a_j = \prod_{i \in C_j} q_i\,. \tag{5.32}$$

Let $S = \sum_{j=1}^{m} Y_j$ and $a = \mathbb{E}[S] = \sum_{j=1}^{m} a_j$. By (5.29) we have $\bar{r} = \mathbb{P}(S > 0)$, and by (5.31) it follows that

$$\bar{r} = \mathbb{E}[S]\,\mathbb{E}\left[\frac{I_{\{S>0\}}}{S} \,\middle|\, Y_J = 1\right] = \mathbb{E}\left[\frac{a}{S} \,\middle|\, Y_J = 1\right],$$

where conditional on $Y_J = 1$ the random variable J takes the value j with probability a_j/a for $j = 1, \ldots, m$. This leads to the following algorithm for estimating the unreliability $\bar{r}$.

Algorithm 5.4.3 (Conditioning via Minimal Cuts)

1. *Generate a discrete random variable J with $\mathbb{P}(J = j) = a_j/a$, $j = 1, \ldots, m$.*
2. *Set X_i equal to 0 for all $i \in C_J$ and generate the values of all other X_i, $i \notin C_J$ from their corresponding* $\mathsf{Ber}(p_i)$ *distributions.*
3. *Evaluate a/S, where S denotes the number of minimal cut sets that have all their components failed (note that $S \geqslant 1$).*
4. *Repeat Steps 1–3 independently N times and take $N^{-1} \sum_{i=1}^{N} a/S_i$ as an estimator of $\bar{r} = \mathbb{P}(S > 0)$.*

It is readily seen that when a, the mean number of failed minimal cuts, is very small, the resulting estimator $\frac{a}{S}$ will have a very small variance. In addition, one can apply importance sampling to the conditional estimator $\frac{a}{S}$ to further reduce the variance.

5.5 STRATIFIED SAMPLING

Stratified sampling is closely related to both the composition method of Section 2.3.3 and the conditional Monte Carlo method discussed in the previous section. As always, we wish to estimate

$$\ell = \mathbb{E}[H(\mathbf{X})] = \int H(\mathbf{x}) f(\mathbf{x})\,\mathrm{d}\mathbf{x}\,.$$

Suppose that $\mathbf{X}$ can be generated via the composition method. Thus, we assume that there exists a random variable Y taking values in $\{1,\ldots,m\}$, say, with known probabilities $\{p_i,\ i=1,\ldots,m\}$, and we assume that it is easy to sample from the conditional distribution of $\mathbf{X}$ given Y. The events $\{Y=i\}, i=1,\ldots,m$ form disjoint subregions, or *strata* (singular: stratum), of the sample space Ω, hence the name *stratification*. Using the conditioning formula (1.11), we can write

$$\ell = \mathbb{E}[\mathbb{E}[H(\mathbf{X}) \mid Y]] = \sum_{i=1}^{m} p_i \, \mathbb{E}[H(\mathbf{X}) \mid Y=i] \,. \tag{5.33}$$

This representation suggests that we can estimate ℓ via the following *stratified sampling estimator*:

$$\widehat{\ell}^{\,s} = \sum_{i=1}^{m} p_i \frac{1}{N_i} \sum_{j=1}^{N_i} H(\mathbf{X}_{ij}) \,, \tag{5.34}$$

where $\mathbf{X}_{ij}$ is the j-th observation from the conditional distribution of $\mathbf{X}$ given $Y=i$. Here N_i is the sample size assigned to the i-th stratum. The variance of the stratified sampling estimator is given by

$$\mathrm{Var}\left(\widehat{\ell}^{\,s}\right) = \sum_{i=1}^{m} \frac{p_i^2}{N_i} \mathrm{Var}(H(\mathbf{X}) \mid Y=i) = \sum_{i=1}^{m} \frac{p_i^2 \, \sigma_i^2}{N_i} \,, \tag{5.35}$$

where $\sigma_i^2 = \mathrm{Var}(H(\mathbf{X}) \mid Y=i)$.

How the strata should be chosen depends very much on the problem at hand. However, for a given particular choice of the strata, the sample sizes $\{N_i\}$ can be obtained in an optimal manner, as given in the next theorem.

Theorem 5.5.1 (Stratified Sampling) *Assuming that a maximum number of N samples can be collected, that is, $\sum_{i=1}^{m} N_i = N$, the optimal value of N_i is given by*

$$N_i^* = N \, \frac{p_i \, \sigma_i}{\sum_{j=1}^{m} p_j \, \sigma_j} \,, \tag{5.36}$$

which gives a minimal variance of

$$Var(\widehat{\ell}^{\,*s}) = \frac{1}{N} \left[\sum_{i=1}^{m} p_i \, \sigma_i \right]^2 \,. \tag{5.37}$$

Proof: The theorem is straightforwardly proved using Lagrange multipliers and is left as an exercise to the reader; see Problem 5.9. □

Theorem 5.5.1 asserts that the minimal variance of $\widehat{\ell}^{\,s}$ is attained for sample sizes N_i that are proportional to $p_i \, \sigma_i$. A difficulty is that although the probabilities p_i are assumed to be known, the standard deviations $\{\sigma_i\}$ are usually unknown. In practice, one would estimate the $\{\sigma_i\}$ from "pilot" runs and then proceed to estimate the optimal sample sizes, N_i^*, from (5.36).

A simple stratification procedure, which can achieve variance reduction without requiring prior knowledge of σ_i^2 and $H(\mathbf{X})$, is presented next.

Proposition 5.5.1 *Let the sample sizes N_i be proportional to p_i, that is, $N_i = p_i N$, $i = 1, \ldots m$. Then*

$$Var(\widehat{\ell}^s) \leqslant Var(\widehat{\ell}) .$$

Proof: Substituting $N_i = p_i N$ in (5.35) yields $\mathrm{Var}(\widehat{\ell}^s) = \frac{1}{N} \sum_{i=1}^m p_i \sigma_i^2$. The result now follows from

$$N \,\mathrm{Var}(\widehat{\ell}) = \mathrm{Var}(H(\mathbf{X})) \geqslant \mathbb{E}[\mathrm{Var}(H(\mathbf{X}) \mid Y)] = \sum_{i=1}^m p_i \sigma_i^2 = N \,\mathrm{Var}(\widehat{\ell}^s),$$

where we have used (5.21) in the inequality. □

Proposition 5.5.1 states that the estimator $\widehat{\ell}^s$ is more accurate than the CMC estimator $\widehat{\ell}$. It effects stratification by favoring those events $\{Y = i\}$ whose probabilities p_i are largest. Intuitively, this cannot, in general, be an optimal assignment, since information on σ_i^2 and $H(\mathbf{X})$ is ignored.

In the special case of equal weights ($p_i = 1/m$ and $N_i = N/m$), the estimator (5.34) reduces to

$$\widehat{\ell}^s = \frac{1}{N} \sum_{i=1}^m \sum_{j=1}^{N/m} H(\mathbf{X}_{ij}) , \tag{5.38}$$

and the method is known as the *systematic sampling method* (see, for example, Cochran [6]).

5.6 IMPORTANCE SAMPLING

The most fundamental variance reduction technique is *importance sampling*. As we shall see below, importance sampling quite often leads to a dramatic variance reduction (sometimes on the order of millions, in particular when estimating rare event probabilities), while with all of the above variance reduction techniques only a moderate reduction, typically up to 10-fold, can be achieved. Importance sampling involves choosing a sampling distribution that favors important samples. Let, as before,

$$\ell = \mathbb{E}_f[H(\mathbf{X})] = \int H(\mathbf{x}) \, f(\mathbf{x}) \, \mathrm{d}\mathbf{x} , \tag{5.39}$$

where H is the sample performance and f is the probability density of $\mathbf{X}$. For reasons that will become clear shortly, we add a subscript f to the expectation to indicate that it is taken with respect to the density f.

Let g be another probability density such that $H\,f$ is *dominated* by g. That is, $g(\mathbf{x}) = 0 \Rightarrow H(\mathbf{x})\, f(\mathbf{x}) = 0$. Using the density g we can represent ℓ as

$$\ell = \int H(\mathbf{x}) \frac{f(\mathbf{x})}{g(\mathbf{x})} \, g(\mathbf{x}) \, \mathrm{d}\mathbf{x} = \mathbb{E}_g \left[H(\mathbf{X}) \frac{f(\mathbf{X})}{g(\mathbf{X})} \right] , \tag{5.40}$$

where the subscript g means that the expectation is taken with respect to g. Such a density is called the *importance sampling* density, *proposal* density, or *instrumental* density (as we use g as an instrument to obtain information about ℓ). Consequently, if $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a *random sample* from g, that is, $\mathbf{X}_1, \ldots, \mathbf{X}_N$ are iid random vectors with density g, then

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^N H(\mathbf{X}_k) \frac{f(\mathbf{X}_k)}{g(\mathbf{X}_k)} \tag{5.41}$$

is an unbiased estimator of ℓ. This estimator is called the *importance sampling estimator*. The ratio of densities,

$$W(\mathbf{x}) = \frac{f(\mathbf{x})}{g(\mathbf{x})}\,, \tag{5.42}$$

is called the *likelihood ratio*. For this reason the importance sampling estimator is also called the *likelihood ratio estimator*. In the particular case where there is no change of measure, that is, $g = f$, we have $W = 1$, and the likelihood ratio estimator in (5.41) reduces to the usual CMC estimator.

5.6.1 Weighted Samples

The likelihood ratios need only be known *up to a constant*, that is, $W(\mathbf{X}) = c\,w(\mathbf{X})$ for some known function $w(\cdot)$. Since $\mathbb{E}_g[W(\mathbf{X})] = 1$, we can write $\ell = \mathbb{E}_g[H(\mathbf{X})\,W(\mathbf{X})]$ as

$$\ell = \frac{\mathbb{E}_g[H(\mathbf{X})\,W(\mathbf{X})]}{\mathbb{E}_g[W(\mathbf{X})]}\,.$$

This suggests, as an alternative to the standard likelihood ratio estimator (5.42), the following *weighted sample estimator*:

$$\widehat{\ell}_w = \frac{\sum_{k=1}^N H(\mathbf{X}_k)\,w_k}{\sum_{k=1}^N w_k}\,. \tag{5.43}$$

Here the $\{w_k\}$, with $w_k = w(\mathbf{X}_k)$, are interpreted as *weights* of the random sample $\{\mathbf{X}_k\}$, and the sequence $\{(\mathbf{X}_k, w_k)\}$ is called a *weighted (random) sample* from $g(\mathbf{x})$. Similar to the regenerative ratio estimator in Chapter 4, the weighted sample estimator (5.43) introduces some bias, which tends to 0 as N increases. Loosely speaking, we may view the weighted sample $\{(\mathbf{X}_k, w_k)\}$ as a representation of $f(\mathbf{x})$ in the sense that $\ell = \mathbb{E}_f[H(\mathbf{X})] \approx \widehat{\ell}_w$ for any function $H(\cdot)$.

5.6.2 The Variance Minimization Method

Since the choice of the importance sampling density g is crucially linked to the variance of the estimator $\widehat{\ell}$ in (5.41), we consider next the problem of minimizing the variance of $\widehat{\ell}$ with respect to g, that is,

$$\min_g \mathrm{Var}_g \left(H(\mathbf{X})\,\frac{f(\mathbf{X})}{g(\mathbf{X})} \right). \tag{5.44}$$

It is not difficult to prove (see, for example, Rubinstein and Melamed [31] and Problem 5.13) that the solution of the problem (5.44) is

$$g^*(\mathbf{x}) = \frac{|H(\mathbf{x})|\,f(\mathbf{x})}{\int |H(\mathbf{x})|\,f(\mathbf{x})\,\mathrm{d}\mathbf{x}}\,. \tag{5.45}$$

In particular, if $H(\mathbf{x}) \geqslant 0$ — which we will assume from now on — then

$$g^*(\mathbf{x}) = \frac{H(\mathbf{x})\,f(\mathbf{x})}{\ell} \tag{5.46}$$

and

$$\mathrm{Var}_{g^*}(\widehat{\ell}) = \mathrm{Var}_{g^*}(H(\mathbf{X})W(\mathbf{X})) = \mathrm{Var}_{g^*}(\ell) = 0\,.$$

The density g^* as per (5.45) and (5.46) is called the *optimal importance sampling density*.

■ **EXAMPLE 5.8**

Let $X \sim \text{Exp}(u^{-1})$ and $H(X) = I_{\{X \geqslant \gamma\}}$ for some $\gamma > 0$. Let f denote the pdf of X. Consider the estimation of

$$\ell = \mathbb{E}_f[H(X)] = \int_\gamma^\infty u^{-1} \mathrm{e}^{-x\,u^{-1}} dx = \mathrm{e}^{-\gamma\,u^{-1}}.$$

We have

$$g^*(x) = H(x)\, f(x)\, \ell^{-1} = I_{\{x \geqslant \gamma\}}\, u^{-1} \mathrm{e}^{-x\,u^{-1}} \mathrm{e}^{\gamma u^{-1}} = I_{\{x \geqslant \gamma\}}\, u^{-1} \mathrm{e}^{-(x-\gamma)\,u^{-1}}.$$

Thus, the optimal importance sampling distribution of X is the *shifted* exponential distribution. Note that Hf is dominated by g^* but f itself is not dominated by g^*. Since g^* is optimal, the likelihood ratio estimator $\widehat{\ell}$ is constant. Namely, with $N = 1$,

$$\widehat{\ell} = H(X)\, W(X) = \frac{H(X) f(X)}{H(X) f(X)/\ell} = \ell\,.$$

It is important to realize that, although (5.41) is an unbiased estimator for *any* pdf g dominating $H\,f$, not all such pdfs are appropriate. One of the main rules for choosing a good importance sampling pdf is that the estimator (5.41) should have finite variance. This is equivalent to the requirement that

$$\mathbb{E}_g\left[H^2(\mathbf{X}) \frac{f^2(\mathbf{X})}{g^2(\mathbf{X})}\right] = \mathbb{E}_f\left[H^2(\mathbf{X}) \frac{f(\mathbf{X})}{g(\mathbf{X})}\right] < \infty\,. \tag{5.47}$$

This suggests that g should not have a "lighter tail" than f and that, preferably, the likelihood ratio, f/g, should be bounded.

In general, implementation of the optimal importance sampling density g^* as per (5.45) and (5.46) is problematic. The main difficulty lies in the fact that to derive $g^*(\mathbf{x})$ one needs to know ℓ. But ℓ is precisely the quantity we want to estimate from the simulation!

In most simulation studies the situation is even worse, since the analytical expression for the sample performance H is unknown in advance. To overcome this difficulty, one can perform a pilot run with the underlying model, obtain a sample $H(\mathbf{X}_1), \ldots, H(\mathbf{X}_N)$, and then use it to estimate g^*. It is important to note that sampling from such an artificially constructed density may be a very complicated and time-consuming task, especially when g is a high-dimensional density.

Remark 5.6.1 (Degeneracy of the Likelihood Ratio Estimator) The likelihood ratio estimator $\widehat{\ell}$ in (5.41) suffers from a form of degeneracy in the sense that the distribution of $W(\mathbf{X})$ under the importance sampling density g may become increasingly skewed as the dimensionality n of $\mathbf{X}$ increases. That is, $W(\mathbf{X})$ may take values close to 0 with high probability, but may also take very large values with a small but significant probability. As a consequence, the variance of $W(\mathbf{X})$ under g may become very large for large n. As an example of this degeneracy, assume for simplicity that the components in $\mathbf{X}$ are iid, under both f and g. Hence, both $f(\mathbf{x})$ and $g(\mathbf{x})$ are the products of their marginal pdfs. Suppose the marginal pdfs of each component X_i are f_1 and g_1, respectively. We can then write $W(\mathbf{X})$ as

$$W(\mathbf{X}) = \exp \sum_{i=1}^n \ln \frac{f_1(X_i)}{g_1(X_i)}\,. \tag{5.48}$$

Using the law of large numbers, the random variable $\sum_{i=1}^{n} \ln\left(f_1(X_i)/g_1(X_i)\right)$ is approximately equal to $n\,\mathbb{E}_{g_1}[\ln\left(f_1(X)/g_1(X)\right)]$ for large n. Hence,

$$W(\mathbf{X}) \approx \exp\left\{-n\,\mathbb{E}_{g_1}\left[\ln\left(\frac{g_1(X)}{f_1(X)}\right)\right]\right\}. \tag{5.49}$$

Since $\mathbb{E}_{g_1}[\ln(g_1(\mathbf{X})/f_1(\mathbf{X}))]$ is nonnegative (see page 31), the likelihood ratio $W(\mathbf{X})$ tends to 0 as $n \to \infty$. However, by definition, the expectation of $W(\mathbf{X})$ under g is always 1. This indicates that the distribution of $W(\mathbf{X})$ becomes increasingly skewed when n gets large. Several methods have been introduced to prevent this degeneracy. Examples are the heuristics of Doucet et al. [8], Liu [23], and Robert and Casella [26] and the so-called screening method. The last will be presented in Sections 5.9 and 8.2.2 and can be considered as a dimension-reduction technique.

When the pdf f belongs to some parametric family of distributions, it is often convenient to choose the importance sampling distribution from the *same* family. In particular, suppose that $f(\cdot) = f(\cdot; \mathbf{u})$ belongs to the family

$$\mathscr{F} = \{f(\cdot; \mathbf{v}),\ \mathbf{v} \in \mathscr{V}\}\,.$$

Then the problem of finding an optimal importance sampling density in this class reduces to the following *parametric* minimization problem:

$$\min_{\mathbf{v}\in\mathscr{V}} \mathrm{Var}_{\mathbf{v}}\left(H(\mathbf{X})\,W(\mathbf{X}; \mathbf{u}, \mathbf{v})\right), \tag{5.50}$$

where $W(\mathbf{X}; \mathbf{u}, \mathbf{v}) = f(\mathbf{X}; \mathbf{u})/f(\mathbf{X}; \mathbf{v})$. We will call the vector $\mathbf{v}$ the *reference parameter vector* or *tilting vector*. Since under $f(\cdot; \mathbf{v})$ the expectation $\ell = \mathbb{E}_{\mathbf{v}}[H(\mathbf{X})\,W(\mathbf{X}; \mathbf{u}, \mathbf{v})]$ is constant, the optimal solution of (5.50) coincides with that of

$$\min_{\mathbf{v}\in\mathscr{V}} V(\mathbf{v})\,, \tag{5.51}$$

where

$$V(\mathbf{v}) = \mathbb{E}_{\mathbf{v}}[H^2(\mathbf{X})\,W^2(\mathbf{X}; \mathbf{u}, \mathbf{v})] = \mathbb{E}_{\mathbf{u}}[H^2(\mathbf{X})\,W(\mathbf{X}; \mathbf{u}, \mathbf{v})]\,. \tag{5.52}$$

We shall call either of the equivalent problems (5.50) and (5.51) the *variance minimization* (VM) problem, and we shall call the parameter vector ${}_*\mathbf{v}$ that minimizes programs (5.50) – (5.51) the *optimal VM reference parameter vector*. We refer to $\mathbf{u}$ as the *nominal* parameter.

The sample average version of (5.51) – (5.52) is

$$\min_{\mathbf{v}\in\mathscr{V}} \widehat{V}(\mathbf{v})\,, \tag{5.53}$$

where

$$\widehat{V}(\mathbf{v}) = \frac{1}{N}\sum_{k=1}^{N}[H^2(\mathbf{X}_k)\,W(\mathbf{X}_k; \mathbf{u}, \mathbf{v})]\,, \tag{5.54}$$

and the sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is from $f(\mathbf{x}; \mathbf{u})$. Note that as soon as the sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is available, the function $\widehat{V}(\mathbf{v})$ becomes a deterministic one.

Since in typical applications both functions $V(\mathbf{v})$ and $\widehat{V}(\mathbf{v})$ are convex and differentiable with respect to $\mathbf{v}$, and since one can typically interchange the expectation and differentiation operators (see Rubinstein and Shapiro [32]), the solutions of programs (5.51) – (5.52) and

(5.53) – (5.54) can be obtained by solving (with respect to $\mathbf{v}$) the following system of equations:

$$\mathbb{E}_{\mathbf{u}}[H^2(\mathbf{X})\,\nabla W(\mathbf{X};\mathbf{u},\mathbf{v})] = \mathbf{0} \tag{5.55}$$

and

$$\frac{1}{N}\sum_{k=1}^{N} H(\mathbf{X}_k)\,\nabla W(\mathbf{X}_k;\mathbf{u},\mathbf{v}) = \mathbf{0}\,, \tag{5.56}$$

respectively, where

$$\nabla W(\mathbf{X};\mathbf{u},\mathbf{v}) = \nabla\frac{f(\mathbf{X};\mathbf{u})}{f(\mathbf{X};\mathbf{v})} = [\nabla \ln f(\mathbf{X};\mathbf{v})]\; W(\mathbf{X};\mathbf{u},\mathbf{v})\,,$$

the gradient is with respect to $\mathbf{v}$ and the function $\nabla \ln f(\mathbf{x};\mathbf{v})$ is the score function, see (1.64). Note that the system of nonlinear equations (5.56) is typically solved using numerical methods.

■ **EXAMPLE 5.9**

Consider estimating $\ell = \mathbb{E}[X]$, where $X \sim \mathsf{Exp}(u^{-1})$. Choosing $f(x;v) = v^{-1}\exp(xv^{-1}), x \geqslant 0$ as the importance sampling pdf, the program (5.51) reduces to

$$\min_v V(v) = \min_v \frac{v}{u^2}\int_0^\infty x^2\,\mathrm{e}^{-(2u^{-1}-v^{-1})x}\,dx = \min_{v\geqslant u/2}\frac{2uv^4}{(2v-u)^3}\,.$$

The optimal reference parameter ${}_*v$ is given by

$${}_*v = 2\,u\,.$$

We see that ${}_*v$ is exactly two times larger than u. Solving the sample average version (5.56) (numerically), one should find that, for large N, its optimal solution ${}_*\widehat{v}$ will be close to the true parameter ${}_*v$.

■ **EXAMPLE 5.10 Example 5.8 (Continued)**

Consider again estimating $\ell = \mathbb{P}_u(X \geqslant \gamma) = \exp(-\gamma u^{-1})$. In this case, using the family $\{f(x;v), v > 0\}$ defined by $f(x;v) = v^{-1}\exp(xv^{-1}), x \geqslant 0$, the program (5.51) reduces to

$$\min_v V(v) = \min_v \frac{v}{u^2}\int_\gamma^\infty \mathrm{e}^{-(2u^{-1}-v^{-1})x}\,dx = \min_{v\geqslant u/2}\frac{v^2}{u}\,\frac{\mathrm{e}^{-\gamma(2u^{-1}-v^{-1})}}{(2v-u)}\,.$$

The optimal reference parameter ${}_*v$ is given by

$${}_*v = \frac{1}{2}\left\{\gamma + u + \sqrt{\gamma^2+u^2}\right\} = \gamma + \frac{u}{2} + \mathcal{O}((u/\gamma)^2)\,,$$

where $\mathcal{O}(x^2)$ is a function of x such that

$$\lim_{x\to 0}\frac{\mathcal{O}(x^2)}{x^2} = \text{constant}\,.$$

We see that for $\gamma \gg u$, ${}_*v$ is approximately equal to γ.

It is important to note that in this case the sample version (5.56) (or (5.53) – (5.54)) is meaningful only for small γ, in particular for those γ for which ℓ is *not a rare-event probability*, say where $\ell < 10^{-5}$. For very small ℓ, a tremendously large sample N is needed (because of the indicator function $I_{\{X \geqslant \gamma\}}$), and thus the importance sampling estimator $\widehat{\ell}$ is useless. We shall discuss the estimation of rare-event probabilities in more detail in Chapter 8.

Observe that the VM problem (5.51) can also be written as

$$\min_{\mathbf{v} \in \mathscr{V}} V(\mathbf{v}) = \min_{\mathbf{v} \in \mathscr{V}} \mathbb{E}_{\mathbf{w}} \left[H^2(\mathbf{X})\, W(\mathbf{X}; \mathbf{u}, \mathbf{v})\, W(\mathbf{X}; \mathbf{u}, \mathbf{w}) \right], \tag{5.57}$$

where $\mathbf{w}$ is an arbitrary reference parameter. Note that (5.57) is obtained from (5.52) by multiplying and dividing the integrand by $f(\mathbf{x}; \mathbf{w})$. We now replace the expected value in (5.57) by its sample (stochastic) counterpart and then take the optimal solution of the associated Monte Carlo program as an estimator of ${}_{*}\mathbf{v}$. Specifically, the stochastic counterpart of (5.57) is

$$\min_{\mathbf{v} \in \mathscr{V}} \widehat{V}(\mathbf{v}) = \min_{\mathbf{v} \in \mathscr{V}} \frac{1}{N} \sum_{k=1}^{N} H^2(\mathbf{X}_k)\, W(\mathbf{X}_k\,; \mathbf{u}, \mathbf{v})\, W(\mathbf{X}_k\,; \mathbf{u}, \mathbf{w}), \tag{5.58}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is an iid sample from $f(\cdot; \mathbf{w})$ and $\mathbf{w}$ is an appropriately chosen *trial* parameter. Solving the stochastic program (5.58) thus yields an estimate, say $\widehat{{}_{*}\mathbf{v}}$, of ${}_{*}\mathbf{v}$. In some cases it may be useful to *iterate* this procedure, that is, use $\widehat{{}_{*}\mathbf{v}}$ as a trial vector in (5.58), to obtain a better estimate.

Once the reference parameter $\mathbf{v} = \widehat{{}_{*}\mathbf{v}}$ is determined, ℓ is estimated via the likelihood ratio estimator

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k)\, W(\mathbf{X}_k; \mathbf{u}, \mathbf{v}), \tag{5.59}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from $f(\cdot; \mathbf{v})$. Typically, the sample size N in (5.59) is larger than that used for estimating the reference parameter. We call (5.59) the *standard likelihood ratio* (SLR) estimator.

5.6.3 The Cross-Entropy Method

An alternative approach for choosing an "optimal" reference parameter vector in (5.59) is based on the Kullback–Leibler cross-entropy, or simply *cross-entropy* (CE), mentioned in (1.59). For clarity we repeat that the CE distance between two pdfs g and h is given (in the continuous case) by

$$\begin{aligned} \mathscr{D}(g, h) &= \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})} \right] = \int g(\mathbf{x}) \ln \frac{g(\mathbf{x})}{h(\mathbf{x})}\, \mathrm{d}\mathbf{x} \\ &= \int g(\mathbf{x}) \ln g(\mathbf{x})\, \mathrm{d}\mathbf{x} - \int g(\mathbf{x}) \ln h(\mathbf{x})\, \mathrm{d}\mathbf{x}\,. \end{aligned} \tag{5.60}$$

Recall that $\mathscr{D}(g, h) \geqslant 0$, with equality if and only if $g = h$.

The general idea is to choose the importance sampling density, say h, such that the CE distance between the optimal importance sampling density g^* in (5.45) and h is minimal. We call this the *CE optimal pdf*. Thus, this pdf solves the following *functional* optimization program:

$$\min_{h} \mathscr{D}(g^*, h)\,.$$

If we optimize over all densities h, then it is immediate from $\mathcal{D}(g^*, h) \geqslant 0$ that the CE optimal pdf coincides with the VM optimal pdf g^*.

As with the VM approach in (5.50) and (5.51), we shall restrict ourselves to the parametric family of densities $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathscr{V}\}$ that contains the "nominal" density $f(\cdot; \mathbf{u})$. The CE method now aims to solve the *parametric* optimization problem

$$\min_{\mathbf{v}} \mathcal{D}\left(g^*, f(\cdot; \mathbf{v})\right).$$

Since the first term on the right-hand side of (5.60) does not depend on $\mathbf{v}$, minimizing the Kullback–Leibler distance between g^* and $f(\cdot; \mathbf{v})$ is equivalent to *maximizing* with respect to $\mathbf{v}$,

$$\int H(\mathbf{x}) f(\mathbf{x}; \mathbf{u}) \ln f(\mathbf{x}; \mathbf{v}) \, d\mathbf{x} = \mathbb{E}_{\mathbf{u}}\left[H(\mathbf{X}) \ln f(\mathbf{X}; \mathbf{v})\right],$$

where we have assumed that $H(\mathbf{x})$ is nonnegative. Arguing as in (5.51), we find that the CE optimal reference parameter vector $\mathbf{v}^*$ can be obtained from the solution of the following simple program:

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{u}}\left[H(\mathbf{X}) \ln f(\mathbf{X}; \mathbf{v})\right]. \tag{5.61}$$

Since typically $D(\mathbf{v})$ is convex and differentiable with respect to $\mathbf{v}$ (see Rubinstein and Shapiro [32]), the solution to (5.61) may be obtained by solving

$$\mathbb{E}_{\mathbf{u}}\left[H(\mathbf{X}) \nabla \ln f(\mathbf{X}; \mathbf{v})\right] = \mathbf{0}, \tag{5.62}$$

provided that the expectation and differentiation operators can be interchanged. The sample counterpart of (5.62) is

$$\frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k) \nabla \ln f(\mathbf{X}_k; \mathbf{v}) = \mathbf{0}. \tag{5.63}$$

By analogy to the VM program (5.51), we call (5.61) the *CE program*, and we call the parameter vector $\mathbf{v}^*$ that minimizes the program (5.64) the *optimal CE reference parameter vector*.

Arguing as in (5.57), it is readily seen that (5.61) is equivalent to the following program:

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{w}}\left[H(\mathbf{X}) W(\mathbf{X}; \mathbf{u}, \mathbf{w}) \ln f(\mathbf{X}; \mathbf{v})\right], \tag{5.64}$$

where $W(\mathbf{X}; \mathbf{u}, \mathbf{w})$ is again the likelihood ratio and $\mathbf{w}$ is an *arbitrary* tilting parameter. Similar to (5.58), we can estimate $\mathbf{v}^*$ as the solution of the stochastic program

$$\max_{\mathbf{v}} \widehat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k) W(\mathbf{X}_k; \mathbf{u}, \mathbf{w}) \ln f(\mathbf{X}_k; \mathbf{v}), \tag{5.65}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from $f(\cdot; \mathbf{w})$. As in the VM case, we mention the possibility of *iterating* this procedure, that is, using the solution of (5.65) as a trial parameter for the next iteration.

Since in typical applications the function $\widehat{D}$ in (5.65) is convex and differentiable with respect to $\mathbf{v}$ (see [32]), the solution of (5.65) may be obtained by solving (with respect to $\mathbf{v}$) the following system of equations:

$$\frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k) W(\mathbf{X}_k; \mathbf{u}, \mathbf{w}) \nabla \ln f(\mathbf{X}_k; \mathbf{v}) = \mathbf{0}, \tag{5.66}$$

where the gradient is with respect to $\mathbf{v}$.

Our extensive numerical studies show that for moderate dimensions n, say $n \leq 50$, the optimal solutions of the CE programs (5.64) and (5.65) (or (5.66)) and their VM counterparts (5.57) and (5.58) are typically nearly the same. However, for high-dimensional problems ($n > 50$), we found numerically that the importance sampling estimator $\widehat{\ell}$ in (5.59) based on VM updating of $\mathbf{v}$ outperforms its CE counterpart in both variance and bias. The latter is caused by the degeneracy of W, to which, we found, CE is more sensitive.

The advantage of the CE program is that it can often be solved *analytically*. In particular, this happens when the distribution of $\mathbf{X}$ belongs to an *exponential family* of distributions; see Section A.3 of the Appendix. Specifically (see (A.16)), for a one-dimensional exponential family parameterized by the mean, the CE optimal parameter is *always*

$$v^* = \frac{\mathbb{E}_u[H(X)\,X]}{\mathbb{E}_u[H(X)]} = \frac{\mathbb{E}_w[W(X;u,w)\,H(X)\,X]}{\mathbb{E}_w[H(X)\,W(X;u,w)]}\,, \tag{5.67}$$

and the corresponding sample-based updating formula is

$$\widehat{v} = \frac{\sum_{k=1}^N H(X_k)\,W(X_k;u,w)\,X_k}{\sum_{k=1}^N H(X_k)\,W(X_k;u,w)}\,, \tag{5.68}$$

respectively, where $X_1,\ldots,X_N$ is a random sample from the density $f(\cdot;w)$ and w is an arbitrary parameter. The multidimensional version of (5.68) is

$$\widehat{v}_i = \frac{\sum_{k=1}^N H(\mathbf{X}_k)\,W(\mathbf{X}_k;\mathbf{u},\mathbf{w})\,X_{ki}}{\sum_{k=1}^N H(\mathbf{X}_k)\,W(\mathbf{X}_k;\mathbf{u},\mathbf{w})} \tag{5.69}$$

for $i = 1,\ldots,n$, where X_{ki} is the i-th component of vector $\mathbf{X}_k$ and $\mathbf{u}$ and $\mathbf{w}$ are parameter vectors.

Observe that for $\mathbf{u} = \mathbf{w}$ (no likelihood ratio term W), (5.69) reduces to

$$\widehat{v}_i = \frac{\sum_{k=1}^N H(\mathbf{X}_k)\,X_{ki}}{\sum_{k=1}^N H(\mathbf{X}_k)}\,, \tag{5.70}$$

where $\mathbf{X}_k \sim f(\mathbf{x};\mathbf{u})$.

Observe also that because of the degeneracy of W, one would always prefer the estimator (5.70) to (5.69), especially for high-dimensional problems. But as we shall see below, this is not always feasible, particularly when estimating rare-event probabilities in Chapter 8.

■ **EXAMPLE 5.11 Example 5.9 continued**

Consider again the estimation of $\ell = \mathbb{E}[X]$, where $X \sim \mathrm{Exp}(u^{-1})$ and $f(x;v) = v^{-1}\exp(xv^{-1}), x \geqslant 0$. Solving (5.62), we find that the optimal reference parameter v^* is equal to

$$v^* = \frac{\mathbb{E}_u[X^2]}{\mathbb{E}_u[X]} = 2\,u\,.$$

Thus, v^* is exactly the same as ${}_*v$. For the sample average of (5.62), we should find that for large N its optimal solution $\widehat{v}^*$ is close to the optimal parameter $v^* = 2u$.

■ **EXAMPLE 5.12 Example 5.10 (Continued)**

Consider again the estimation of $\ell = \mathbb{P}_u(X \geqslant \gamma) = \exp(-\gamma u^{-1})$. In this case, we readily find from (5.67) that the optimal reference parameter is $v^* = \gamma + u$. Note that similar to the VM case, for $\gamma \gg u$, the optimal reference parameter is approximately γ.

Note that in the above example, similar to the VM problem, the CE sample version (5.66) is meaningful only when γ is chosen such that ℓ is *not a rare-event probability*, say when $\ell < 10^{-4}$. In Chapter 8 we present a general procedure for estimating rare-event probabilities of the form $\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma)$ for an arbitrary function $S(\mathbf{x})$ and level γ.

■ **EXAMPLE 5.13 Finite Support Discrete Distributions**

Let X be a discrete random variable with finite support, that is, X can only take a finite number of values, say $a_1, \ldots, a_m$. Let $u_i = \mathbb{P}(X = a_i), i = 1, \ldots, m$ and define $\mathbf{u} = (u_1, \ldots, u_m)$. The distribution of X is thus trivially parameterized by the vector $\mathbf{u}$. We can write the density of X as

$$f(x; \mathbf{u}) = \sum_{i=1}^{m} u_i \, I_{\{x=a_i\}} \,.$$

From the discussion at the beginning of this section we know that the optimal CE and VM parameters *coincide*, since we optimize over *all* densities on $\{a_1, \ldots, a_m\}$. By (5.45) the VM (and CE) optimal density is given by

$$\begin{aligned} f(x; \mathbf{v}^*) &= \frac{H(x)\, f(x; \mathbf{u})}{\sum_x H(x)\, f(x; \mathbf{u})} \\ &= \frac{\sum_{i=1}^m H(a_i)\, u_i \, I_{\{x=a_i\}}}{\mathbb{E}_{\mathbf{u}}[H(X)]} \\ &= \sum_{i=1}^{m} \frac{H(a_i)\, u_i}{\mathbb{E}_{\mathbf{u}}[H(X)]} \, I_{\{x=a_i\}} \\ &= \sum_{i=1}^{m} \left(\frac{\mathbb{E}_u[H(X)\, I_{\{X=a_i\}}]}{\mathbb{E}_{\mathbf{u}}[H(X)]} \right) I_{\{x=a_i\}} \,, \end{aligned}$$

so that

$$v_i^* = \frac{\mathbb{E}_{\mathbf{u}}[H(X)\, I_{\{X=a_i\}}]}{\mathbb{E}_{\mathbf{u}}[H(X)]} = \frac{\mathbb{E}_{\mathbf{w}}[H(X)\, W(X; \mathbf{u}, \mathbf{w}) I_{\{X=a_i\}}]}{\mathbb{E}_{\mathbf{w}}[H(X)\, W(X; \mathbf{u}, \mathbf{w})]} \tag{5.71}$$

for any reference parameter $\mathbf{w}$, provided that $\mathbb{E}_{\mathbf{w}}[H(X)\, W(X; \mathbf{u}, \mathbf{w})] > 0$.

The vector $\mathbf{v}^*$ can be estimated from the stochastic counterpart of (5.71), that is, as

$$\widehat{v}_i = \frac{\sum_{k=1}^N H(X_k)\, W(X_k; \mathbf{u}, \mathbf{w}) I_{\{X_k=a_i\}}}{\sum_{k=1}^N H(X_k)\, W(X_k; \mathbf{u}, \mathbf{w})} \,, \tag{5.72}$$

where $X_1, \ldots, X_N$ is an iid sample from the density $f(\cdot; \mathbf{w})$.

A similar result holds for a random vector $\mathbf{X} = (X_1, \ldots, X_n)$ where $X_1, \ldots, X_n$ are independent discrete random variables with finite support, characterized by

the parameter vectors $\mathbf{u}_1, \ldots, \mathbf{u}_n$. Because of the independence assumption, the CE problem (5.64) separates into n subproblems of the form above, and all the components of the optimal CE reference parameter $\mathbf{v}^* = (\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*)$, which is now a vector of vectors, follow from (5.72). Note that in this case the optimal VM and CE reference parameters are usually not equal, since we are not optimizing the CE over all densities. See, however, Proposition 4.2 in Rubinstein and Kroese [29] for an important case where they *do* coincide and yield a zero-variance likelihood ratio estimator.

The updating rule (5.72), which involves discrete finite support distributions, and in particular the Bernoulli distribution, will be extensively used for combinatorial optimization problems later on in the book.

■ **EXAMPLE 5.14 Example 5.1 (Continued)**

Consider the bridge network in Figure 5.1, and let

$$S(\mathbf{X}) = \min\{X_1 + X_4,\ X_1 + X_3 + X_5,\ X_2 + X_3 + X_4,\ X_2 + X_5\}\,.$$

Suppose we wish to estimate the probability that the shortest path from node A to node B has a length of at least γ; that is, with $H(\mathbf{x}) = I_{\{S(\mathbf{x}) \geqslant \gamma\}}$, we want to estimate

$$\ell = \mathbb{E}[H(\mathbf{X})] = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}]\,.$$

We assume that the components $\{X_i\}$ are independent, that $X_i \sim \mathsf{Exp}(u_i^{-1})$, $i = 1, \ldots, 5$, and that γ is chosen such that $\ell \geq 10^{-2}$. Thus, here the CE updating formula (5.69) and its particular case (5.70) (with $\mathbf{w} = \mathbf{u}$) applies. We shall show that this yields substantial variance reduction. The likelihood ratio in this case is

$$\begin{aligned} W(\mathbf{x}; \mathbf{u}, \mathbf{v}) &= \frac{f(\mathbf{x}; \mathbf{u})}{f(\mathbf{x}; \mathbf{v})} = \frac{\prod_{i=1}^5 \frac{1}{u_i} \mathrm{e}^{-x_i/u_i}}{\prod_{i=1}^5 \frac{1}{v_i} \mathrm{e}^{-x_i/v_i}} \\ &= \exp\left(-\sum_{i=1}^5 x_i \left(\frac{1}{u_i} - \frac{1}{v_i}\right)\right) \prod_{i=1}^5 \frac{v_i}{u_i}\,. \end{aligned} \tag{5.73}$$

As a concrete example, let the *nominal* parameter vector $\mathbf{u}$ be equal to $(1, 1, 0.3, 0.2, 0.1)$ and let $\gamma = 1.5$. We will see that this probability ℓ is approximately 0.06.

Note that the typical length of a path from A to B is smaller than $\gamma = 1.5$; hence, using importance sampling instead of CMC should be beneficial. The idea is to estimate the optimal parameter vector $\mathbf{v}^*$ *without* using likelihood ratios, that is, using (5.70), since likelihood ratios, as in (5.69) (with quite arbitrary $\mathbf{w}$, say by guessing an initial trial vector $\mathbf{w}$), would typically make the estimator of $\mathbf{v}^*$ unstable, especially for high-dimensional problems.

Denote by $\widehat{\mathbf{v}}_1$ the CE estimator of $\mathbf{v}^*$ obtained from (5.70). We can iterate (repeat) this procedure, say for T iterations, using (5.69), and starting with $\mathbf{w} = \widehat{\mathbf{v}}_1, \widehat{\mathbf{v}}_2, \ldots$. Once the final reference vector $\widehat{\mathbf{v}}_T$ is obtained, we then estimate ℓ via a *larger* sample from $f(\mathbf{x}; \widehat{\mathbf{v}}_T)$, say of size N_1, using the SLR estimator (5.59). Note, however, that for high-dimensional problems, iterating in this way could lead to an unstable final estimator $\widehat{\mathbf{v}}_T$. In short, a single iteration with (5.70) might often be the best alternative.

Table 5.1 presents the performance of the estimator (5.59), starting from $\mathbf{w} = \mathbf{u} = (1, 1, 0.3, 0.2, 0.1)$ and then iterating (5.69) three times. Note again that in the first iteration we generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{u})$ and then apply (5.70) to obtain an estimate $\widehat{\mathbf{v}} = (\widehat{v}_1, \ldots, \widehat{v}_5)$ of the CE optimal reference parameter vector $\mathbf{v}^*$. The sample sizes for updating $\widehat{\mathbf{v}}$ and calculating the estimator $\widehat{\ell}$ were $N = 10^3$ and $N_1 = 10^5$, respectively. In the table RE denotes the estimated relative error.

Table 5.1 Iterating the five-dimensional vector $\widehat{\mathbf{v}}$.

iteration	$\widehat{\mathbf{v}}$					$\widehat{\ell}$	RE
0	1	1	0.3	0.2	0.1	0.0643	0.0121
1	2.4450	2.3274	0.2462	0.2113	0.1030	0.0631	0.0082
2	2.3850	2.3894	0.3136	0.2349	0.1034	0.0644	0.0079
3	2.3559	2.3902	0.3472	0.2322	0.1047	0.0646	0.0080

Note that $\widehat{\mathbf{v}}$ already converged after the first step, so using likelihood ratios in Steps 2 and 3 did not add anything to the quality of $\widehat{\mathbf{v}}$. It also follows from the results of Table 5.1 that CE outperforms CMC (compare the relative errors 0.008 and 0.0121 for CE and CMC, respectively). To obtain a similar relative error of 0.008 with CMC would require a sample size of approximately $2.5 \cdot 10^5$ instead of 10^5; we thus obtained a reduction by a factor of 2.5 when using the CE estimation procedure. As we shall see in Chapter 8 for smaller probabilities, a variance reduction of several orders of magnitude can be achieved.

5.7 SEQUENTIAL IMPORTANCE SAMPLING

Sequential importance sampling (SIS), also called *dynamic importance sampling*, is simply importance sampling carried out in a sequential manner. To explain the SIS procedure, consider the expected performance ℓ in (5.39) and its likelihood ratio estimator $\widehat{\ell}$ in (5.41), with $f(\mathbf{x})$ the "target" and $g(\mathbf{x})$ the importance sampling, or proposal, pdf. Suppose that (a) $\mathbf{X}$ is decomposable, that is, it can be written as a vector $\mathbf{X} = (X_1, \ldots, X_n)$, where each of the X_i may be multi-dimensional, and (b) it is easy to sample from $g(\mathbf{x})$ sequentially. Specifically, suppose that $g(\mathbf{x})$ is of the form

$$g(\mathbf{x}) = g_1(x_1)\, g_2(x_2 \,|\, x_1) \cdots g_n(x_n \,|\, x_1, \ldots, x_{n-1}), \tag{5.74}$$

where it is easy to generate X_1 from density $g_1(x_1)$, and conditional on $X_1 = x_1$, the second component from density $g_2(x_2|x_1)$, and so on, until one obtains a single random vector $\mathbf{X}$ from $g(\mathbf{x})$. Repeating this independently N times, each time sampling from $g(\mathbf{x})$, one obtains a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $g(\mathbf{x})$ and estimates ℓ according to (5.41). To further simplify the notation, we abbreviate $(x_1, \ldots, x_t)$ to $\mathbf{x}_{1:t}$ for all t. In particular, $\mathbf{x}_{1:n} = \mathbf{x}$. Typically, t can be viewed as a (discrete) time parameter and $\mathbf{x}_{1:t}$ as a path or trajectory. By the product rule of probability (1.4), the target pdf $f(\mathbf{x})$ can also be written sequentially, that is,

$$f(\mathbf{x}) = f(x_1)\, f(x_2 \,|\, x_1) \cdots f(x_n \,|\, \mathbf{x}_{1:n-1}). \tag{5.75}$$

From (5.74) and (5.75) it follows that we can write the likelihood ratio in product form as

$$W(\mathbf{x}) = \frac{f(x_1)\, f(x_2 \mid x_1) \cdots f(x_n \mid \mathbf{x}_{1:n-1})}{g_1(x_1)\, g_2(x_2 \mid x_1) \cdots g_n(x_n \mid \mathbf{x}_{1:n-1})} \tag{5.76}$$

or, if $W_t(\mathbf{x}_{1:t})$ denotes the likelihood ratio up to time t, recursively as

$$W_t(\mathbf{x}_{1:t}) = u_t\, W_{t-1}(\mathbf{x}_{1:t-1}), \quad t = 1, \ldots, n\,, \tag{5.77}$$

with initial weight $W_0(\mathbf{x}_{1:0}) = 1$ and *incremental weights* $u_1 = f(x_1)/g_1(x_1)$ and

$$u_t = \frac{f(x_t \mid \mathbf{x}_{1:t-1})}{g_t(x_t \mid \mathbf{x}_{1:t-1})} = \frac{f(\mathbf{x}_{1:t})}{f(\mathbf{x}_{1:t-1})\, g_t(x_t \mid \mathbf{x}_{1:t-1})}\,, \quad t = 2, \ldots, n\,. \tag{5.78}$$

In order to update the likelihood ratio recursively, as in (5.78), one needs to known the marginal pdfs $f(\mathbf{x}_{1:t})$. This may not be easy when f does not have a Markov structure, as it requires integrating $f(\mathbf{x})$ over all $x_{t+1}, \ldots, x_n$. Instead one can introduce a sequence of *auxiliary* pdfs $f_1, f_2, \ldots, f_n$ that are easily evaluated and such that each $f_t(\mathbf{x}_{1:t})$ is a good approximation to $f(\mathbf{x}_{1:t})$. The terminating pdf f_n must be equal to the original f. Since

$$f(\mathbf{x}) = \frac{f_1(x_1)}{1} \frac{f_2(\mathbf{x}_{1:2})}{f_1(x_1)} \cdots \frac{f_n(\mathbf{x}_{1:n})}{f_{n-1}(\mathbf{x}_{1:n-1})}\,, \tag{5.79}$$

we have as a generalization of (5.78) the incremental updating weight

$$u_t = \frac{f_t(\mathbf{x}_{1:t})}{f_{t-1}(\mathbf{x}_{1:t-1})\, g_t(x_t \mid \mathbf{x}_{1:t-1})} \tag{5.80}$$

for $t = 1, \ldots, n$, where we put $f_0(\mathbf{x}_{1:0}) = 1$.

Remark 5.7.1 Note that the incremental weights u_t only need to be defined *up to a constant*, say c_t, for each t. In this case the likelihood ratio $W(\mathbf{x})$ is known up to a constant as well, say $W(\mathbf{x}) = C\, w(\mathbf{x})$, where $1/C = \mathbb{E}_g[w(\mathbf{X})]$ can be estimated via the corresponding sample mean. In other words, when the normalization constant is unknown, one can still estimate ℓ using the weighted sample estimator (5.43) rather than the likelihood ratio estimator (5.42).

Summarizing, the SIS method can be written as follows.

Algorithm 5.7.1 (SIS Algorithm)

1. *For each finite $t = 1, \ldots, n$, sample X_t from $g_t(x_t \mid \mathbf{x}_{1:t-1})$.*

2. *Compute $w_t = u_t\, w_{t-1}$, where $w_0 = 1$ and*

$$u_t = \frac{f_t(\mathbf{X}_{1:t})}{f_{t-1}(\mathbf{X}_{1:t-1})\, g_t(X_t \mid \mathbf{X}_{1:t-1})}, \quad t = 1, \ldots, n. \tag{5.81}$$

3. *Repeat N times and estimate ℓ via $\widehat{\ell}$ in (5.42) or $\widehat{\ell}_w$ in (5.43).*

■ **EXAMPLE 5.15 Random Walk on the Integers**

Consider the random walk on the integers of Example 1.10 on page 19, with probabilities p and q for jumping up or down, respectively. Suppose that $p < q$, so that the walk has a drift toward $-\infty$. Our goal is to estimate the rare-event probability ℓ of reaching state K before state 0, starting from state $0 < k \ll K$, where K is a large number. As an intermediate step consider first the probability of reaching K in exactly n steps, that is, $\mathbb{P}(X_n = K) = \mathbb{E}[I_{A_n}]$, where $A_n = \{X_n = K\}$. We have

$$f(\mathbf{x}_{1:n}) = f(x_1 \mid k)\, f(x_2 \mid x_1)\, f(x_3 \mid x_2) \ldots f(x_n \mid x_{n-1}) \,,$$

where the conditional probabilities are either p (for upward jumps) or q (for downward jumps). If we simulate the random walk with *different* upward and downward probabilities, $\tilde{p}$ and $\tilde{q}$, then the importance sampling pdf $g(\mathbf{x}_{1:n})$ has the same form as $f(\mathbf{x}_{1:n})$ above. Thus, the importance weight after Step t is updated via the incremental weight

$$u_t = \frac{f(x_t \mid x_{t-1})}{g(x_t \mid x_{t-1})} = \begin{cases} p/\tilde{p} & \text{if } x_t = x_{t-1} + 1 \,, \\ q/\tilde{q} & \text{if } x_t = x_{t-1} - 1 \,. \end{cases}$$

The probability $\mathbb{P}(A_n)$ can now be estimated via importance sampling as

$$\frac{1}{N} \sum_{i=1}^{N} W_{i,n}\, I_{\{X_{i,n}=K\}} \,, \tag{5.82}$$

where the paths $\mathbf{X}_{i,1:n}, i = 1, \ldots, N$ are generated via g, rather than f and $W_{i,n}$ is the likelihood ratio of the i-th such path. Returning to the estimation of ℓ, let τ be the first time that either 0 or K is reached. Writing $I_{\{X_t=K\}} = H(\mathbf{X}_{1:t})$, we have

$$\begin{aligned} \ell &= \mathbb{E}_f[I_{\{X_\tau=K\}}] = \mathbb{E}_f[H(\mathbf{X}_{1:\tau})] = \sum_{n=1}^{\infty} \mathbb{E}[H(\mathbf{X}_{1:n})\, I_{\{\tau=n\}}] \\ &= \sum_{n=1}^{\infty} \sum_{\mathbf{x}} H(\mathbf{x}_{1:n})\, I_{\{\tau=n\}}\, f(\mathbf{x}_{1:n}) \\ &= \sum_{n=1}^{\infty} \sum_{\mathbf{x}} \underbrace{\frac{f(\mathbf{x}_{1:n})}{g(\mathbf{x}_{1:n})} I_{\{x_n=K\}}}_{\tilde{H}(\mathbf{x}_{1:n})}\, I_{\{\tau=n\}}\, g(\mathbf{x}_{1:n}) \\ &= \mathbb{E}_g[\tilde{H}(\mathbf{X}_{1:\tau})] = \mathbb{E}_g[W_\tau\, I_{\{X_\tau=K\}}] \,, \end{aligned}$$

with W_τ the likelihood ratio of $\mathbf{X}_{1:\tau}$, which can be updated at each time t by multiplying with either $p/\tilde{p}$ or $q/\tilde{q}$ for upward and downward steps, respectively. Note that $I_{\{\tau=n\}}$ is indeed a function of $\mathbf{x}_n = (x_1, \ldots, x_n)$. This leads to the same estimator as (5.82) with the deterministic n replaced by the stochastic τ. It can be shown (see, for example, [5]) that choosing $\tilde{p} = q$ and $\tilde{q} = p$, that is, *interchanging* the probabilities, gives an efficient estimator for ℓ.

5.7.1 Nonlinear Filtering for Hidden Markov Models

This section describes an application of SIS to nonlinear filtering. Many problems in engineering, applied sciences, statistics, and econometrics can be formulated as *hidden Markov models* (HMM). In its simplest form, an HMM is a stochastic process $\{(X_t, Y_t)\}$ where X_t (which may be multidimensional) represents the *true* state of some system and Y_t represents the *observed* state of the system at a discrete time t. It is usually assumed that $\{X_t\}$ is a Markov chain, say with initial distribution $f(x_0)$ and one-step transition probabilities $f(x_t \mid x_{t-1})$. It is important to note that the actual state of the Markov chain remains *hidden*, hence the name HMM. All information about the system is conveyed by the process $\{Y_t\}$. We assume that, given $X_0, \ldots, X_t$, the observation Y_t depends only on X_t via some conditional pdf $f(y_t \mid x_t)$. Note that we have used here a Bayesian style of notation in which all (conditional) probability densities are represented by the *same symbol* f. We will use this notation throughout the rest of this section. We denote by $\mathbf{X}_{1:t} = (X_1, \ldots, X_t)$ and $\mathbf{Y}_{1:t} = (Y_1, \ldots, Y_t)$ the unobservable and observable sequences up to time t, respectively — and similarly for their lowercase equivalents.

The HMM is represented graphically in Figure 5.2. This is an example of a *Bayesian network*. The idea is that edges indicate the dependence structure between two variables. For example, given the states $X_1, \ldots, X_t$, the random variable Y_t is conditionally independent of $X_1, \ldots, X_{t-1}$, because there is no direct edge from Y_t to any of these variables. We thus have $f(y_t \mid \mathbf{x}_{1:t}) = f(y_t \mid x_t)$, and more generally

$$f(\mathbf{y}_{1:t} \mid \mathbf{x}_{1:t}) = f(y_1 \mid x_1) \cdots f(y_t \mid x_t) = f(\mathbf{y}_{1:t-1} \mid \mathbf{x}_{1:t-1})\, f(y_t \mid x_t)\,. \tag{5.83}$$

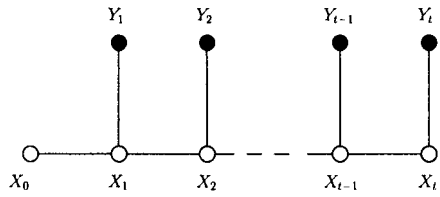


Figure 5.2 A graphical representation of the HMM

Summarizing, we have

$$\begin{aligned} X_t &\sim f(x_t \mid x_{t-1}) - \text{ state equation} \\ Y_t &\sim f(y_t \mid x_t) - \text{ observation equation.} \end{aligned} \tag{5.84}$$

■ **EXAMPLE 5.16**

An example of (5.84) is the following popular model:

$$\begin{aligned} X_t &= \varphi_1(X_{t-1}) + \varepsilon_{1t} \\ Y_t &= \varphi_2(X_t) + \varepsilon_{2t}\,, \end{aligned} \tag{5.85}$$

where $\varphi_1(\cdot)$ and $\varphi_2(\cdot)$ are given vector functions and ε_{1t} and ε_{2t} are independent d-dimensional Gaussian random vectors with zero mean and covariance matrices C_1 and C_2, respectively.

Our goal, based on an outcome $\mathbf{y}_{1:t}$ of $\mathbf{Y}_{1:t}$, is to determine, or estimate *on-line*, the following quantities:

1. The joint conditional pdf $f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ and, as a special case, the marginal conditional pdf $f(x_t \mid \mathbf{y}_{1:t})$, which is called the *filtering* pdf.

2. The expected performance

$$\ell = \mathbb{E}_{f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})}[H(\mathbf{X}_{1:t})] = \int H(\mathbf{x}_{1:t})\, f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})\, \mathrm{d}\mathbf{x}_{1:t} \,. \tag{5.86}$$

It is well known [8] that the conditional pdf $f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ or the filtering pdf $f(x_t \mid \mathbf{y}_{1:t})$ can be found explicitly only for the following two particular cases:

(a) When $\varphi_1(x)$ and $\varphi_2(x)$ in (5.85) are linear, the filtering pdf is obtained from the celebrated *Kalman filter*. The Kalman filter is explained in Section A.6 of the Appendix.

(b) When the $\{x_t\}$ can take only a finite number, say K, of possible values, for example, as in binary signals, one can calculate $f(x_t \mid \mathbf{y}_{1:t})$ efficiently with complexity $\mathcal{O}(K^2 t)$. Applications can be found in digital communication and speech recognition; see, for example, Section A.7 of the Appendix.

Because the target pdf $f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ for the general state space model (5.84) is difficult to obtain exactly, one needs to resort to Monte Carlo methods. To put the nonlinear filtering problem in the sequential Monte Carlo framework of Section 5.7, we first write $f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:n})$ in sequential form, similar to (5.79). A natural candidate for the "auxiliary" pdf at time t is the conditional pdf $f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$. That is, only the observations up to time t are used. By Bayes' rule we have for each $t = 1, \ldots, n$,

$$\begin{aligned}
&\frac{f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})}{f(\mathbf{x}_{1:t-1} \mid \mathbf{y}_{1:t-1})} \\
&= \frac{f(\mathbf{y}_{1:t} \mid \mathbf{x}_{1:t}) f(\mathbf{x}_{1:t})}{f(\mathbf{y}_{1:t})} \frac{f(\mathbf{y}_{1:t-1})}{f(\mathbf{y}_{1:t-1} \mid \mathbf{x}_{1:t-1}) f(\mathbf{x}_{1:t-1})} \\
&= \frac{f(\mathbf{y}_{1:t-1} \mid \mathbf{x}_{1:t-1})\, f(y_t \mid x_t)\, f(\mathbf{x}_{1:t-1})\, f(x_t \mid x_{t-1})}{f(\mathbf{y}_{1:t-1})\, f(y_t \mid \mathbf{y}_{1:t-1})} \frac{f(\mathbf{y}_{1:t-1})}{f(\mathbf{y}_{1:t-1} \mid \mathbf{x}_{1:t-1}) f(\mathbf{x}_{1:t-1})} \\
&= \frac{f(y_t \mid x_t)\, f(x_t \mid x_{t-1})}{f(y_t \mid \mathbf{y}_{1:t-1})} \,,
\end{aligned} \tag{5.87}$$

where we have also used (5.83) and the fact that $f(x_t \mid \mathbf{x}_{1:t-1}) = f(x_t \mid x_{t-1})$, $t = 1, 2, \ldots$ by the Markov property.

This result is of little use for an exact calculation of $f(\mathbf{x}_{1:n} \mid \mathbf{y}_{1:n})$, since it requires computation of $f(y_t \mid \mathbf{y}_{1:t-1})$, which involves the evaluation of complicated integrals. However, if both functions (pdfs) $f(x_t \mid x_{t-1})$ and $f(y_t \mid x_t)$ can be evaluated exactly (which is a reasonable assumption), then SIS can be used to approximately simulate from $f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ as follows: Let $g_t(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ be the importance sampling pdf. We assume that, similar to (5.74), we can write $g_t(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ recursively as

$$g_t(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t}) = g_0(x_0 \mid y_0) \prod_{s=1}^{t} g_s(x_s \mid \mathbf{x}_{s-1}, \mathbf{y}_s) \,. \tag{5.88}$$

Then, by analogy to (5.77), and using (5.87) (dropping the normalization constant $f(y_t \mid \mathbf{y}_{1:t-1})$), we can write the importance weight w_t of a path $\mathbf{x}_{1:t}$ generated from $g_t(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ recursively as

$$w_t = w_{t-1} \frac{f(y_t \mid x_t)\, f(x_t \mid x_{t-1})}{g_t(x_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t})} = w_{t-1}\, u_t \,. \tag{5.89}$$

A natural choice for the importance sampling pdf is

$$g_t(x_t \mid \mathbf{x}_{1:t-1}, \mathbf{y}_{1:t}) = f(x_t \mid x_{t-1}) \,, \tag{5.90}$$

in which case the incremental weight simplifies to

$$u_t = f(y_t \mid x_t). \tag{5.91}$$

With this choice of sampling distribution, we are simply guessing the values of the hidden process $\{X_t\}$ without paying attention to the observed values.

Once the importance sampling density is chosen, sampling from the target pdf $f(\mathbf{x}_{1:t} \mid \mathbf{y}_{1:t})$ proceeds as described in Section 5.7. For more details, the interested reader is referred to [8], [23], and [26].

■ **EXAMPLE 5.17 Bearings-Only Tracking**

Suppose we want to track an object (e.g., a submarine) via a radar device that only reports the *angle* to the object (see Figure 5.3). In addition, the angle measurements are noisy. We assume that the initial position and velocity are known and that the object moves at a constant speed.

Let $X_t = (p_{1t}, v_{1t}, p_{2t}, v_{2t})^T$ be the vector of positions and (discrete) velocities of the target object at time $t = 0, 1, 2, \ldots$, and let Y_t be the measured angle. The problem is to track the unknown state of the object X_t based on the measurements $\{Y_t\}$ and the initial conditions.

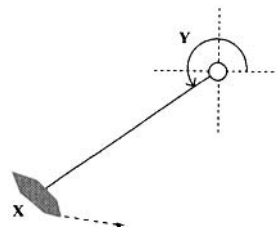


Figure 5.3 Track the object via noisy measurements of the angle.

The process $(X_t, Y_t), t = 0, 1, 2, \ldots$ is described by the following system:

$$\begin{aligned} X_t &= A\, X_{t-1} + \varepsilon_{1t} \\ Y_t &= \arctan(p_{1t}, p_{2t}) + \varepsilon_{2t} \,. \end{aligned}$$

Here $\arctan(u, v)$ denotes the four-quadrant arc-tangent, that is, $\arctan(v/u) + c$, where c is either 0, $\pm\pi$, or $\pm\pi/2$, depending on the quadrant in which (u, v) lies.

The random noise vectors $\{\varepsilon_{1t}\}$ are assumed to be $\mathsf{N}(\mathbf{0}, C_1)$ distributed, and the measurement noise ε_{2t} is $\mathsf{N}(0, \sigma_2^2)$ distributed. All noise variables are independent of each other. The matrix A is given by

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The problem is to find the conditional pdf $f(x_t \mid \mathbf{y}_{1:t})$ and, in particular, the expected system state $\mathbb{E}[X_t \mid \mathbf{y}_{1:t}]$.

We indicate how this problem can be solved via SIS. Using (5.90) for the sampling distribution means simply that X_t is drawn from a $\mathsf{N}(Ax_{t-1}, C_1)$ distribution. As a consequence of (5.91) the incremental weight, $u_t = f(y_t \mid x_t)$, is equal to the value at y_t of the normal pdf with mean $\arctan(p_{1t}, p_{2t})$ and variance σ_2^2. The corresponding SIS procedure is summarized below. We note that the SIS procedure is often implemented in parallel; that is, instead of computing the $\{w_{kt}\}$ and $\{x_{kt}\}$ in series, one can compute them at the same time by running N parallel processes.

SIS Procedure

1. Initialize X_0.
2. For each $t = 1, \ldots, n$ draw $X_t \sim \mathsf{N}(AX_{t-1}, C_1)$.
3. Update the weights $w_t = u_t\, w_{t-1}$, where $w_0 = 1$ and

$$u_t = \frac{1}{\sigma_2 \sqrt{2\pi}} \exp\left\{ -\frac{1}{2} \left(\frac{y_t - \arctan(p_{1t}, p_{2t})}{\sigma_2} \right)^2 \right\}.$$

4. Repeat N times and estimate the expected system state at time t as

$$\widehat{x}_t = \frac{\sum_{k=1}^N w_{kt}\, x_{kt}}{\sum_{k=1}^N w_{kt}},$$

where x_{kt} and w_{kt} are the state and weight for the k-th sample, respectively.

As a numerical illustration, consider the case where $\sigma_2 = 0.005$ and

$$C_1 = \sigma_1^2 \begin{pmatrix} 1/4 & 1/2 & 0 & 0 \\ 1/2 & 1 & 0 & 0 \\ 0 & 0 & 1/4 & 1/2 \\ 0 & 0 & 1/2 & 1 \end{pmatrix},$$

with $\sigma_1 = 0.001$. Let $X_0 \sim \mathsf{N}(\boldsymbol{\mu}_0, \Sigma_0)$, with $\boldsymbol{\mu}_0 = (-0.05, 0.001, 0.2, -0.055)^T$, and

$$\Sigma_0 = 0.1^2 \begin{pmatrix} 0.5^2 & 0 & 0 & 0 \\ 0 & 0.005^2 & 0 & 0 \\ 0 & 0 & 0.3^2 & 0 \\ 0 & 0 & 0 & 0.01^2 \end{pmatrix}.$$

Figure 5.4 shows how the estimated process $\{\widehat{x}_t\}$ tracks the actual process $\{x_t\}$ over 100 time steps.

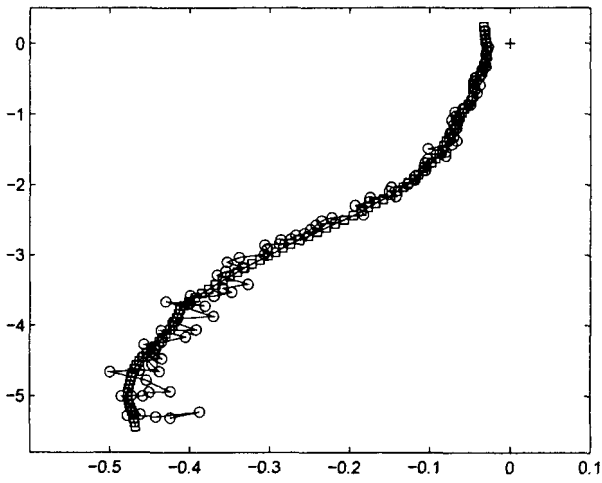


Figure 5.4 Tracking with SIS.

As time increases, the tracking rapidly becomes more unstable. This is a consequence of the degeneracy of the likelihood ratio. Indeed, after a few iterations, only a handful of samples contain the majority of the importance weight. This yields high variability between many runs and provides less reliable estimates. To prevent this degeneracy, several heuristic resampling techniques have been proposed; see, for example, [8].

5.8 THE TRANSFORM LIKELIHOOD RATIO METHOD

The *transform likelihood ratio* (TLR) method is a simple, convenient, and *unifying* way of constructing efficient importance sampling estimators. To motivate the TLR method, we consider the estimation of

$$\ell = \mathbb{E}[H(\mathbf{X})] \,, \tag{5.92}$$

where $\mathbf{X} \sim f(\mathrm{x})$. Consider first the case where $\mathbf{X}$ is one-dimensional (we write X instead of $\mathbf{X}$). Let F be the cdf of X. According to the IT method, we can write

$$X = F^{-1}(U) \,, \tag{5.93}$$

where $U \sim \mathsf{U}(0,1)$ and F^{-1} is the inverse of the cdf F. Substituting $X = F^{-1}(U)$ into $\ell = \mathbb{E}[H(X)]$, we obtain

$$\ell = \mathbb{E}[H(F^{-1}(U))] = \mathbb{E}[\widetilde{H}(U)] \,.$$

Note that in contrast to $\ell = \mathbb{E}[H(X)]$, where the expectation is taken with respect to $f(x)$, in $\ell = [\widetilde{H}(U)]$, the expectation is taken with respect to the uniform $\mathsf{U}(0,1)$ distribution. The extension to the multidimensional case is simple.

Let $h(u;\nu)$ be another density on $(0,1)$, parameterized by some reference parameter ν, with $h(u;\nu) > 0$ for all $0 \leqslant u \leqslant 1$ (note that u is a variable and not a parameter). An example is the Beta$(\nu, 1)$ distribution, with density

$$h(u;\nu) = \nu\, u^{\nu-1}, \quad u \in (0,1)\,,$$

with $\nu > 0$, or the Beta$(1,\nu)$ distribution, with density

$$h(u;\nu) = \nu\,(1-u)^{\nu-1}, \quad u \in (0,1)\,.$$

Using Beta$(1,\nu)$ as the importance sampling pdf, we can write ℓ as

$$\ell = \mathbb{E}_\nu[\widetilde{H}(U)\,\widetilde{W}(U;\nu)]\,, \tag{5.94}$$

where $U \sim h(u;\nu)$, and

$$\widetilde{W}(U;\nu) = \frac{1}{h(U;\nu)} \tag{5.95}$$

is the likelihood ratio. The likelihood ratio estimator of ℓ is given by

$$\widehat{\ell} = N^{-1} \sum_{k=1}^{N} \widetilde{H}(U_k)\,\widetilde{W}(U_k;\nu)\,, \tag{5.96}$$

where $U_1, \ldots, U_N$ is a random sample from $h(u;\nu)$, We call (5.96) the *inverse transform likelihood ratio* (ITLR) estimator; see Kroese and Rubinstein [19].

Suppose, for example, $X \sim$ Weib(α,λ), that is, X has the density

$$f(x;\alpha,\lambda) = \alpha\lambda(\lambda x)^{\alpha-1}\mathrm{e}^{-(\lambda x)^\alpha}. \tag{5.97}$$

Note that a Weibull random variable can be generated using the transformation

$$X = \lambda^{-1}\, Z^{1/\alpha}, \tag{5.98}$$

where Z is a random variable distributed Exp(1). Applying the IT method, we obtain

$$X = F^{-1}(U) = \lambda^{-1}(-\ln(1-U))^{1/\alpha}, \tag{5.99}$$

and $\widetilde{H}(U_i)\,\widetilde{W}(U_i;\nu)$ in (5.96) reduces to $H(\lambda^{-1}\,(-\ln(1-U_i))^{1/\alpha})/h(U_i;\nu)$.

The TLR method is a natural extension of the ITLR method. It comprises two steps. The first is a simple *change of variable* step, and the second involves an application of the SLR technique to the transformed pdf.

To apply the first step, we simply write $\mathbf{X}$ as a function of another random vector, say as

$$\mathbf{X} = G(\mathbf{Z})\,. \tag{5.100}$$

If we define

$$\widetilde{H}(\mathbf{Z}) = H(G(\mathbf{Z}))\,,$$

then estimating (5.92) is equivalent to estimating

$$\ell = \mathbb{E}[\widetilde{H}(\mathbf{Z})]\,. \tag{5.101}$$

Note that the expectations in (5.92) and (5.101) are taken with respect to the original density of $\mathbf{X}$ and the transformed density of $\mathbf{Z}$. As an example, consider again a one-dimensional

case and let $X \sim \text{Weib}(\alpha, \lambda)$. Recalling (5.98), we have $\widetilde{H}(Z) = H(\lambda^{-1} Z^{1/\alpha})$ and thus, $\ell = \mathbb{E}[H(\lambda^{-1} Z^{1/\alpha})]$.

To apply the second step, we assume that $\mathbf{Z}$ has a density $h(\mathbf{z}; \boldsymbol{\theta})$ in some class of densities $\{h(\mathbf{z}; \boldsymbol{\eta})\}$. Then we can seek to estimate ℓ efficiently via importance sampling, for example, using the standard likelihood ratio method. In particular, by analogy to (5.59), we obtain the following estimator:

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^{N} \widetilde{H}(\mathbf{Z}_k) \, \widetilde{W}(\mathbf{Z}_k; \boldsymbol{\theta}, \boldsymbol{\eta}) \,, \tag{5.102}$$

where

$$\widetilde{W}(\mathbf{Z}_k; \boldsymbol{\theta}, \boldsymbol{\eta}) = \frac{h(\mathbf{Z}_k; \boldsymbol{\theta})}{h(\mathbf{Z}_k; \boldsymbol{\eta})}$$

and $\mathbf{Z}_k \sim h(\mathbf{z}; \boldsymbol{\eta})$. We shall call the SLR estimator (5.102) based on the transformation (5.100), the *TLR estimator*. As an example, consider again the $\text{Weib}(\alpha, \lambda)$ case. Using (5.98), we could take $h(z; \eta) = \eta \, \mathrm{e}^{-\eta z}$ as the sampling pdf, with $\eta = \theta = 1$ as the nominal parameter. Hence, in this case, $\widehat{\ell}$ in (5.102) reduces to

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^{N} \widetilde{H}\left(\lambda^{-1} Z_k^{1/\alpha}\right) \widetilde{W}(Z_k; \theta, \eta) \,, \tag{5.103}$$

with

$$\widetilde{W}(Z_k; \theta, \eta) = \frac{h(Z_k; \theta)}{h(Z_k; \eta)} = \frac{\theta \, \mathrm{e}^{-\theta Z_k}}{\eta \, \mathrm{e}^{-\eta Z_k}}$$

and $Z_k \sim \text{Exp}(\eta)$.

To find the optimal parameter vector $\boldsymbol{\eta}^*$ of the TLR estimator (5.102) we can solve, by analogy to (5.64), the following CE program:

$$\max_{\boldsymbol{\eta}} D(\boldsymbol{\eta}) = \max_{\boldsymbol{\eta}} \mathbb{E}_{\boldsymbol{\tau}} \left[\widetilde{H}(\mathbf{Z}) \, \widetilde{W}(\mathbf{Z}; \boldsymbol{\theta}, \boldsymbol{\tau}) \ln h(\mathbf{Z}; \boldsymbol{\eta}) \right] \tag{5.104}$$

and similarly for the stochastic counterpart of (5.104).

Since $\mathbf{Z}$ can be distributed quite arbitrarily, one would typically choose its distribution from an exponential family of distributions (see Section A.3 of the Appendix), for which the optimal solution $\boldsymbol{\eta}^*$ of (5.104) can be obtained analytically in a convenient and simple form. Below we present the TLR algorithm for estimating $\ell = \mathbb{E}_f[H(\mathbf{X})]$, assuming that $\mathbf{X}$ is a random vector with independent, continuously distributed components.

Algorithm 5.8.1 (TLR Algorithm)

1. *For a given random vector* $\mathbf{X}$*, find a transformation* G *such that* $\mathbf{X} = G(\mathbf{Z})$*, with* $\mathbf{Z} \sim h(\mathbf{z}; \boldsymbol{\theta})$*. For example, take* $\mathbf{Z}$ *with all components being iid and distributed according to an exponential family (e.g.,* $\text{Exp}(1)$*).*

2. *Generate a random sample* $\mathbf{Z}_1, \ldots, \mathbf{Z}_N$ *from* $h(\cdot; \boldsymbol{\tau})$*.*

3. *Solve the stochastic counterpart of the program* (5.104) *(for a one-parameter exponential family parameterized by the mean, apply directly the analytic solution (A.18)). Iterate if necessary. Denote the solution by* $\widehat{\boldsymbol{\eta}}$*.*

4. *Generate a (larger) random sample* $\mathbf{Z}_1, \ldots, \mathbf{Z}_{N_1}$ *from* $h(\cdot; \widehat{\eta})$ *and estimate* $\ell = \mathbb{E}[H(G(\mathbf{Z}))]$ *via the TLR estimator* (5.102)*, taking* $\eta = \widehat{\eta}$.

The TLR Algorithm 5.8.1 ensures that as soon as the transformation $\mathbf{X} = G(\mathbf{Z})$ is chosen, one can estimate ℓ using the TLR estimator (5.102) instead of the SLR estimator (5.59). Although the accuracy of both estimators (5.102) and (5.59) is the same (Rubinstein and Kroese [29]), the advantage of the former is its universality and it ability to avoid the computational burden while directly delivering the analytical solution of the stochastic counterpart of the program (5.104).

5.9 PREVENTING THE DEGENERACY OF IMPORTANCE SAMPLING

In this section, we show how to prevent the *degeneracy* of importance sampling estimators. The degeneracy of likelihood ratios in high-dimensional Monte Carlo simulation problems is one of the central topics in Monte Carlo simulation. To prevent degeneracy, several heuristics have been introduced (see, for example, [8], [23], [26]), which are not widely used by the Monte Carlo community. In this section we first present a method introduced in [22] called the *screening* method. Next, we present its new modification, which quite often allows substantial reduction of the dimension of the likelihood ratios. By using this modification we not only automatically prevent the degeneracy of importance sampling estimators but also obtain variance reduction.

To motivate the screening method, consider again Example 5.14 and observe that only the first two importance sampling parameters of the five-dimensional vector $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{v}_2, \widehat{v}_3, \widehat{v}_4, \widehat{v}_5)$ are substantially different from those in the nominal parameter vector $\mathbf{u} = (u_1, u_2, u_3, u_4, u_5)$. The reason is that the partial derivatives of ℓ with respect to u_1 and u_2 are significantly larger than those with respect to u_3, u_4, and u_5. We call such elements u_1 and u_2 *bottleneck elements*. Based on this observation, one could use instead of the importance sampling vector $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{v}_2, \widehat{v}_3, \widehat{v}_4, \widehat{v}_5)$ the vector $\widehat{\mathbf{v}} = (\widehat{v}_1, \widehat{v}_2, u_3, u_4, u_5)$, reducing the number of importance sampling parameters from five to two. This not only has computational advantages — one needs to solve a two-dimensional variance or CE minimization program instead of a five-dimensional one — but also leads to further variance reduction since the likelihood ratio term W with two product terms is less "noisy" than the one with five product terms.

To identify the bottleneck elements in our bridge example we need to estimate, for every $i = 1, \ldots, 5$, the partial derivative

$$\begin{aligned}
\frac{\partial}{\partial u_i} \ell(\mathbf{u}) &= \int I_{\{S(\mathbf{x}) \geqslant \gamma\}} \frac{\partial}{\partial u_i} f(\mathbf{x}; \mathbf{u})\, \mathrm{d}\mathbf{x} \\
&= \int I_{\{S(\mathbf{x}) \geqslant \gamma\}} \frac{\frac{\partial}{\partial u_i} f(\mathbf{x}; \mathbf{u})}{f(\mathbf{x}; \mathbf{u})} f(\mathbf{x}; \mathbf{u})\, \mathrm{d}\mathbf{x} \\
&= \mathbb{E}_{\mathbf{u}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma\}} \frac{\partial}{\partial u_i} \ln f(\mathbf{X}; \mathbf{u}) \right] \\
&= \mathbb{E}_{\mathbf{u}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma\}} \left(-\frac{1}{u_i} + \frac{X_i}{u_i^2} \right) \right] .
\end{aligned}$$

Observe that for general $\mathbf{w} \neq \mathbf{u}$ we obtain

$$\frac{\partial}{\partial u_i} \ell(\mathbf{u}) = \mathbb{E}_{\mathbf{w}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma\}} \left(-\frac{1}{u_i} + \frac{X_i}{u_i^2} \right) W(\mathbf{X}; \mathbf{u}, \mathbf{w}) \right] .$$

Point estimators for $\partial \ell(\mathbf{u})/\partial u_i$, based on a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x}; \mathbf{v})$, are thus found as the sample mean, say M, of the random variables

$$I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} \left(-\frac{1}{u_i} + \frac{X_{ki}}{u_i^2} \right) W(\mathbf{X}_k; \mathbf{u}, \mathbf{v}), \quad k = 1, \ldots, N . \tag{5.105}$$

The corresponding $(1-\alpha)$ confidence interval is given (see (4.7)) by $(M \pm z_{1-\alpha/2}\, S/\sqrt{N})$, with S being the sample standard deviation of (5.105).

Table 5.2 presents point estimates and 95% confidence intervals for $\partial \ell(\mathbf{u})/\partial u_i, i = 1, \ldots, 5$, with $\ell(\mathbf{u}) = \mathbb{P}(S(\mathbf{X}) \geqslant 1.5)$ and $\mathbf{u} = (1, 1, 0.3, 0.2, 0.1)$, as in Example 5.14. The partial derivatives were estimated using the initial parameter vector $\mathbf{u} = (1, 1, 0.3, 0.2, 0.1)$ and a sample size of $N = 10^4$. The bottleneck cuts, corresponding to the largest values of the partial derivatives, are marked in the last column of the table by asterisks.

Table 5.2 Point estimates and confidence intervals for $\partial \ell(\mathbf{u})/\partial u_i, i = 1, \ldots, 5$.

i	PE	CI	bottleneck
1	8.7 e–2	(7.8 e–2, 9.5 e–2)	*
2	9.2 e–2	(8.3 e–2, 1.0 e–1)	*
3	– 6.0 e–3	(–2.2 e–2, 1.0 e–2)	
4	4.5 e–2	(1.5 e–2, 7.4 e–2)	
5	5.5 e–2	(6.4 e–4, 1.1 e–1)	

We see in Table 5.2 that not only are the partial derivatives with respect to the first two components larger than the remaining three, but also that the variability in the estimates is much smaller for the first two. So, we can exclude the remaining three from updating and thus proceed with the first two.

Table 5.3 presents data similar to those in Table 5.1 using the screening method, that is, with $(v_1, v_2, u_3, u_4, u_5)$ starting from $\mathbf{u} = (1, 1, 0.3, 0.2, 0.1)$ and iterating again (5.70) (for the first iteration) and (5.69) two more times. One can see that the results are very similar to the ones obtained in Table 5.1.

Table 5.3 Iterating the two-dimensional vector $\widehat{\mathbf{v}}$ using the screening method.

iteration	$\widehat{\mathbf{v}}$					$\widehat{\ell}$	RE
0	1	1	0.3	0.2	0.1	0.0623	0.0123
1	2.1790	2.5119	0.3	0.2	0.1	0.0641	0.0079
2	2.3431	2.4210	0.3	0.2	0.1	0.0647	0.0080
3	2.3407	2.2877	0.3	0.2	0.1	0.0642	0.0079

In general, large-dimensional, complex simulation models contain both bottleneck and nonbottleneck parameters. The number of bottleneck parameters is typically smaller than the number of nonbottleneck parameters. Imagine a situation where the size (dimension)

of the vector $\mathbf{u}$ is large, say 100, and the number of bottleneck elements is only about 10–15. Then, clearly, an importance sampling estimator based on bottleneck elements alone will not only be much more accurate than its standard importance sampling counterpart involving all 100 likelihood ratios (containing both bottleneck and nonbottleneck ones), but in contrast to the latter will not be degenerated.

The bottleneck phenomenon often occurs when one needs to estimate the probability of a nontypical event in the system, like a rare-event probability. This will be treated in Chapter 8. For example, if one observes a failure in a reliability system with highly reliable elements, then it is very likely that several elements (typically the less reliable ones) forming a minimal cut in the model all fail simultaneously. Another example is the estimation of a buffer overflow probability in a queueing network, that is, the probability that the total number of customers in all queues exceeds some large number. Again, if a buffer overflow occurs, it is quite likely that this has been caused by a buildup in the bottleneck queue, which is the most congested one in the network.

Recall that for high-dimensional simulation models the CE updating formula (5.65) is useless, since the likelihood ratio term W is the product of a large number of marginal likelihoods, and will cause degeneracy and large variance of the resulting importance sampling estimator $\widehat{\ell}$. On the other hand, importance sampling combined with screening and involving only a relatively small number of bottleneck elements (and thus a product of a relatively small number of likelihoods) will not only lead to tremendous variance reduction but will produce a stable estimator as well.

If not stated otherwise, we will deal with estimating the following performance:

$$\ell = \ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})] \,,$$

where $H(\mathbf{X})$ is assumed to be an arbitrary sample function and $\mathbf{X}$ is an n-dimensional random vector with pdf $f(\mathbf{x};\mathbf{u})$. A particular case is $H(\mathbf{X}) = I_{\{S(\mathbf{X})\geq\gamma\}}$, that is, $H(\mathbf{X})$ is an indicator function. In Chapter 8 we apply our methodology to rare events, that is, we assume that γ is very large, so ℓ is a rare-event probability, say $\ell \leq 10^{-6}$.

Next, we introduce a modification of the above screening method [22], where we screen out the bottleneck parameters using the estimator $\widehat{\mathbf{v}}$ in (5.70) rather than (the estimator of) the gradient of $\ell(\mathbf{u})$. As we shall see below, there are certain advantages in using the vector $\widehat{\mathbf{v}}$ for identifying the bottleneck parameters rather than its gradient.

5.9.1 The Two-Stage Screening Algorithm

Here we present a two-stage screening algorithm, where at the first stage we *identify* the bottleneck parameters and at the second stage we *find* the estimator of the optimal bottleneck parameter vector by solving the standard convex CE program (5.66). For simplicity, we assume that the components of $\mathbf{X}$ are independent and that each component is distributed according to a one-dimensional exponential family that is parameterized by the mean — the dependent case could be treated similarly. Moreover, $H(\mathbf{x})$ is assumed to be a monotonically increasing function in each component of $\mathbf{x}$. A consequence of the above assumptions is that the parameter vector $\mathbf{v}$ has dimension n. That is, $\mathbf{v} = (v_1, \ldots, v_n)$.

Let $B \subset \{1, \ldots, n\}$ denote the indices of the bottleneck parameters and $\bar{B}$ denote the indices of the nonbottleneck parameters. For any n-dimensional vector $\mathbf{y}$, let $\mathbf{y}_V$ denote the $|V|$-dimensional vector with components $\{y_i, i \in V\}$.

As soon as B is identified in the first stage and the corresponding optimal parameter vector, $\mathbf{v}_B^*$ say, is estimated in the second stage, via $\widehat{\mathbf{v}}_B$ say, one can estimate ℓ via

$$\widehat{\ell}_B = \frac{1}{N} \sum_{k=1}^{N} H(\mathbf{X}_k)\, W_B(\mathbf{X}_{kB}; \mathbf{u}_B, \widehat{\mathbf{v}}_B) \,, \tag{5.106}$$

where $\mathbf{X}_{kB}$ is the k-th sample of $\mathbf{X}_B$,

$$W_B(\mathbf{X}_B; \mathbf{u}_B, \widehat{\mathbf{v}}_B) = \frac{f_B(\mathbf{X}_B; \mathbf{u}_B)}{f_B(\mathbf{X}_B; \widehat{\mathbf{v}}_B)} \,,$$

and f_B is the pdf of $\mathbf{X}_B$. We call this estimator the *screening estimator*.

Note that $\mathbf{u}$ and the nonscreening importance sampling estimator $\widehat{\mathbf{v}}$ of the optimal parameter $\mathbf{v}^*$ can be written as $\mathbf{u} = (\mathbf{u}_B, \mathbf{u}_{\bar{B}})$ and $\widehat{\mathbf{v}} = (\widehat{\mathbf{v}}_B, \widehat{\mathbf{v}}_{\bar{B}})$, where $\mathbf{u}_{\bar{B}}$ and $\widehat{\mathbf{v}}_{\bar{B}}$ denote the *nonbottleneck* parameter vectors of the original and estimated reference parameter vectors of the standard importance sampling estimator (5.59), respectively. It is crucial to understand that in the screening estimator (5.106) we automatically set $\widehat{\mathbf{v}}_{\bar{B}} = \mathbf{u}_{\bar{B}}$. Consequently, because of the independence of the components, the likelihood ratio term $W(\mathbf{X})$ reduces to a product of $|B|$ quotients of marginal pdfs instead of the product of n such quotients. Note also that the optimal parameter vector $\widehat{\mathbf{v}}_B$ in (5.106) can be obtained by using the standard convex CE program (5.65), provided that the pair $(\mathbf{u}, \mathbf{v})$ is replaced by its bottleneck counterpart $(\mathbf{u}_B, \mathbf{v}_B)$.

For convenience we rewrite (5.65) by omitting W (that is, $\mathbf{X}_1, \ldots, \mathbf{X}_N$ are generated under $\mathbf{u}$). We have

$$\max_{\mathbf{v}} \widehat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i)\, \ln f(\boldsymbol{X}_i; \mathbf{v}) \,. \tag{5.107}$$

Since (5.107) contains no likelihood ratios, the parameter vector $\widehat{\mathbf{v}}$ obtained from the solution of (5.107) should be quite accurate.

We shall implement screening for both CE and VM methods. Recall that for the CE method the parameter vector $\widehat{\mathbf{v}}$ (and $\widehat{\mathbf{v}}_B$) can often be updated analytically, in particular when the sampling distribution comes from an exponential family. In contrast, for VM the updating typically involves a numerical procedure.

The identification of the size of the bottleneck parameter vector $\mathbf{v}_B^*$ at the first stage is associated with a simple classification procedure that divides the n-dimensional vector $\widehat{\mathbf{v}}$ into two parts, namely, $\widehat{\mathbf{v}} = (\widehat{\mathbf{v}}_B, \widehat{\mathbf{v}}_{\bar{B}})$, such that $\widehat{\mathbf{v}}_{\bar{B}} \approx \mathbf{u}_{\bar{B}}$ (componentwise), while $\widehat{\mathbf{v}}_B \neq \mathbf{u}_B$ (componentwise). Note that it can be readily shown, by analogy to Proposition A.4.2 in the Appendix, that if each component of the random vector $\mathbf{X}$ is from a one-parameter exponential family parameterized by the mean and if $H(\mathbf{x})$ is a monotonically increasing function in each component of $\mathbf{x}$, then each element of $\mathbf{v}_B^*$ is at least as large as the corresponding one of $\mathbf{u}_B$. We leave the proof as an exercise for the reader.

Below we present a detailed two-stage screening algorithm based on CE, denoted as the CE-SCR algorithm. Its VM counterpart, the VM-SCR algorithm, is similar. At the first stage of our algorithm we purposely solve the same program (5.107) several times. By doing so we collect statistical data, which are used to identify the size of the estimator of $\mathbf{v}_B^*$.

Algorithm 5.9.1 (CE-SCR Two-Stage Screening Algorithm)

1. *Initialize the set of bottleneck elements to* $B_0 = \{1, \ldots, n\}$. *Set* $t = 1$.

2. *Generate a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *from* $f(\mathbf{x}; \mathbf{u})$ *and deliver the CE solution of the stochastic program* (5.107). *Denote the solution by* $\widehat{\mathbf{v}}_t = (\widehat{v}_{t1}, \ldots, \widehat{v}_{tn})$. *Note that* $\widehat{\mathbf{v}}_t$ *is an* n*-dimensional parameter vector.*

3. *Calculate the relative perturbation for each element* $\widehat{v}_{ti}$, $i = 1, \ldots, n$ *as*

$$\delta_{ti} = \frac{\widehat{v}_{ti} - u_i}{u_i} . \tag{5.108}$$

4. *If* $\delta_{ti} < \delta$, *where* δ *is some threshold value, say* $\delta = 0.1$ *(note that negative* δ_{ti} *automatically satisfies* $\delta_{ti} < \delta$*), set* $\widehat{v}_{ti} = u_i$, *that is, identify the* i*-th element of the vector* $\mathbf{v}$ *as a nonbottleneck parameter. Otherwise, identify it as a bottleneck one. Let* B_t *be the set of bottleneck elements at iteration* t.

5. *Repeat Steps 2–4 several times, say* d *times, each time increasing* t *by 1 and updating the set* B_t. *Note that the sequence of sets* B_t, $t = 1, \ldots, d$ *is nonincreasing.*

6. *Apply the standard CE program (5.107) to estimate the optimal parameter vector* $\widehat{\mathbf{v}}_B$, *with* $B = B_d$. *Deliver (5.106) as the resulting estimator of the rare-event probability* ℓ.

It is important to note the following:

1. As mentioned, under the present assumptions (independent components, each from a one-parameter exponential family parameterized by the mean, and $H(\mathbf{x})$ monotonically increasing in each component), the components of the $\mathbf{v}^*$ are at least as large as the corresponding elements of $\mathbf{u}$. Taking this into account, Algorithm 5.9.1 always identifies all elements i corresponding to $\delta_i < 0$ as nonbottleneck ones.

2. Recall that Steps 2–4 are purposely performed d times. This allows one to better determine the nonbottleneck parameters, since it is likely that they will fluctuate around their nominal value u_i and therefore δ_i will become negative or very small in one of the replications.

3. The advantage of Algorithm 5.9.1 compared to its gradient counterpart is that identification of the bottleneck elements in the former is based on the relative perturbations δ_i (see (5.108)) with respect to the *known* original parameter values u_i, while in the latter it is based on the absolute value of the gradient itself. It is not difficult to see that the former classifier, the so-called $\widehat{\mathbf{v}}$-based classifier, is more natural to use than the gradient-based one. In addition, we found numerically that it identifies more accurately the actual bottleneck size.

5.9.1.1 Numerical Results We next present numerical studies with Algorithm 5.9.1 for a generalization of the bridge system in Example 5.1, depicted in Figure 5.5.

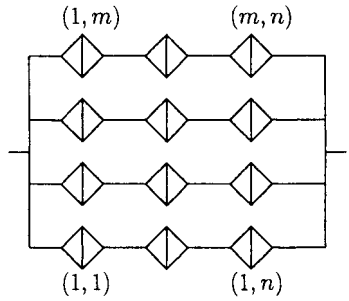


Figure 5.5 An $m \times n$ bridge system.

The system consists of $m \times n$ bridges arranged in a grid, and all bridges are of the form in Figure 5.1. Denote the lengths of the edges within the (i, j)-th bridge by $X_{ij1}, \ldots, X_{ij5}$. Then the length of the shortest path through bridge (i, j) is

$$Y_{ij} = \min\{X_{ij1} + X_{ij4},\ X_{ij2} + X_{ij5},\ X_{ij1} + X_{ij3} + X_{ij5},\ X_{ij2} + X_{ij3} + X_{ij4}\}\ . \tag{5.109}$$

Suppose we want to estimate the expected maximal length ℓ of the shortest paths in all rows, that is, $\ell = \mathbb{E}[H(\mathbf{X})]$, with

$$H(\mathbf{X}) = \max\{Y_{11} + \cdots + Y_{1n}, \ldots, Y_{m1} + \cdots + Y_{mn}\}\ . \tag{5.110}$$

In our numerical results, we assume that the components X_{ijk} of the random vector $\mathbf{X}$ are independent and that each component has a Weib(α, u) distribution, that is, X_{ijk} has the density

$$f(x; \alpha, u) = \alpha u (ux)^{\alpha - 1} \mathrm{e}^{-(ux)^{\alpha}},$$

with $u = u_{ijk}$. Recall that such a Weibull random variable can be generated using the transformation $X = u^{-1} Z^{1/\alpha}$, where Z is a random variable distributed Exp(1). We also assume that only u is controllable, while α is fixed and equals 0.2. We purposely selected some elements of $\mathbf{u}$ to be bottleneck ones and set $\delta = 0.1$. It is important to realize that the $\{X_{ijk}\}$ are here *not* parameterized by the mean. However, by taking $1/u_{ijk}$ as the parameter, we are in the framework described above. In particular, the relative perturbations are carried out with respect to $\alpha/\widehat{v}_{ijk}$ and α/u_{ijk}.

Table 5.4 presents the performance of Algorithm 5.9.1 for the 1×1 (single-bridge) model (5.110). Here $u_{111} = 1$ and $u_{112} = 1$ are chosen to be the bottleneck parameters, whereas the remaining (nonbottleneck) ones are set equal to 2. The notations in Table 5.4 are as follows:

1. *Mean*, *max*, and *min* $\widehat{\ell}$ denote the sample mean, maximum, and minimum values of 10 independently generated estimates of $\widehat{\ell}$.

2. *RE* denotes the sample relative error for $\widehat{\ell}$, averaged over the 10 runs.

3. *CPU* denotes the average CPU time in seconds based on 10 runs.

Table 5.4 Performance of Algorithm 5.9.1 for the single-bridge model with samples $N = N_1 = 500$.

	CMC	CE	VM	CE-SCR	VM-SCR
Mean $\widehat{\ell}$	4.052	3.970	3.734	3.894	3.829
Max $\widehat{\ell}$	8.102	4.327	4.201	4.345	4.132
Min $\widehat{\ell}$	1.505	3.380	3.395	3.520	3.278
RE	0.519	0.070	0.078	0.076	0.068
CPU	0.00	0.04	0.21	0.05	0.13

From the results of Table 5.4, it follows that for this relatively small model both CE and VM perform similarly to their screening counterparts. We will further see (see also Chapter 8) that as the complexity of the model increases, VM-SCR outperforms its three alternatives, in particular CE-SCR. Note also that for this model, both CE and VM detected correctly at the first stage the two bottleneck parameters. In particular, Table 5.5 presents a typical dynamics of detecting the two bottleneck parameters at the first stage of Algorithm 5.9.1 for a single-bridge model having a total of 5 parameters. In Table 5.5, t denotes the replication number at the first stage, while the 0s and 1s indicate whether the corresponding parameters are identified as nonbottleneck or bottleneck parameters, respectively. One can see that after two replications four bottleneck parameters are left, after six replications three are identified as bottleneck parameters, and after seven replications the process stabilizes, detecting correctly the two true bottleneck parameters.

Table 5.5 Typical dynamics for detecting the bottleneck parameters at the first stage of Algorithm 5.9.1 for the bridge model.

t	u_1	u_2	u_3	u_4	u_5
0	1	1	1	1	1
1	1	1	0	1	1
2	1	1	0	1	1
3	1	1	0	1	0
4	1	1	0	1	0
5	1	1	0	1	0
6	1	1	0	1	0
7	1	1	0	0	0
8	1	1	0	0	0
9	1	1	0	0	0

Table 5.6 presents a typical evolution of the sequence $\{\widehat{\mathbf{v}}_t\}$ in the single-bridge model for the VM and VM-SCR methods at the second stage of Algorithm 5.9.1.

Table 5.6 Typical evolution of the sequence $\{\widehat{\mathbf{v}}_t\}$ for the VM and VM-SCR methods.

	VM						VM-SCR				
t	$\widehat{v}_1$	$\widehat{v}_2$	$\widehat{v}_3$	$\widehat{v}_4$	$\widehat{v}_5$	t	$\widehat{v}_1$	$\widehat{v}_2$	$\widehat{v}_3$	$\widehat{v}_4$	$\widehat{v}_5$
	1.000	1.000	2.000	2.000	2.000		1.000	1.000	2	2	2
1	0.537	0.545	2.174	2.107	1.615	1	0.555	0.599	2	2	2
2	0.346	0.349	2.071	1.961	1.914	2	0.375	0.402	2	2	2
3	0.306	0.314	1.990	1.999	1.882	3	0.315	0.322	2	2	2

One can clearly see that the bottleneck parameters decrease about three times after the third iteration, while the nonbottleneck ones fluctuate about their nominal value $u = 2$.

Table 5.7 presents the performance of Algorithm 5.9.1 for the 3×10 bridge model with six bottlenecks corresponding to the elements u_{111}, u_{112}, u_{211}, u_{212}, u_{311}, u_{312}. We set $u_{111} = u_{112} = u_{211} = u_{212} = u_{311} = u_{312} = 1$, while the remaining (nonbottlenecks) values are set equal to 2. Note again that in this case both CE and VM found the true six bottlenecks.

Table 5.7 Performance of Algorithm 5.9.1 for the 3×10 model with six bottleneck elements and sample size $N = N_1 = 1000$.

	CMC	CE	VM	CE-SCR	VM-SCR
Mean $\widehat{\ell}$	16.16	16.11	14.84	16.12	15.67
Max $\widehat{\ell}$	22.65	26.85	16.59	18.72	17.20
Min $\widehat{\ell}$	11.13	7.007	12.59	14.63	14.80
RE	0.20	0.34	0.075	0.074	0.049
CPU	0.00	0.49	68.36	0.73	27.54

From the results in Table 5.7, it follows that without screening even the naive Monte Carlo outperforms the standard CE. However, using screening results in substantial improvement of CE. Finally, VM-SCR outperforms all four remaining alternatives.

Other successful applications of the screening method will be given in Chapter 8 when dealing with estimations of probabilities of rare events for complex high-dimensional simulation models. In the following subsection we present a case study using the screening method.

5.9.2 Case Study

In this section we present a case study with the screening method applied to the so-called CONFTRA (composite generation/transmission) reliability model applied to the Brazilian electric power system [15]. Our representation closely follows [21].

The generating system of the CONFTRA model comprises n_g generating units, while the transmission system consists of n_l load/generation busses connected by lines. Each generator and transmission line is assumed to be either functioning (1) or failed (0). The component states are gathered into the binary random vector $\mathbf{X} = (X_1, \ldots, X_{n_g}, X_{n_g+1}, \ldots, X_{n_g+n_l})$. It is assumed that $X_i \sim \mathsf{Ber}(u_i)$, $i = 1, \ldots, n_g + n_l$ and that the $\{X_i\}$ are mutually independent. Thus, the joint pdf $f(\mathbf{x}; \mathbf{u})$ can be written as

$$f(\mathbf{x}; \mathbf{u}) = \prod_{i=1}^{n_g+n_l} f_i(x_i; u_i), \quad \text{with } f_i(x_i; u_i) = u_i^{x_i}(1-u_i)^{1-x_i}, \quad x_i \in \{0, 1\}. \quad (5.111)$$

The analysis of the system is based on the linearized power flow model [15]. If the power flow in one or more circuits exceeds the respective maximum flow capacity, the system will be overloaded. The goal is to estimate the *expected unserved energy* (EUE) in the system. For a given state vector $\mathbf{x}$ the unserved energy, $H(\mathbf{x})$, corresponds to the minimum load curtailments required to relieve the system overloads. For each scenario $\mathbf{x}$, the unserved

energy $H(\mathbf{x})$ is obtained from the solution of some linear programming problem, the details of which are not important here (but see [15]). The EUE can thus be written as $\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})]$, and the corresponding CMC estimator is $\widehat{\ell}(\mathbf{u}) = \frac{1}{N}\sum_{k=1}^{N} H(\mathbf{X}_k)$, where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from $f(\mathbf{x}; \mathbf{u})$, and for each $\mathbf{X}_k$, $k = 1, \ldots, N$ the function $H(\mathbf{X}_k)$ is obtained from the solution of a linear program.

Using the CMC estimator is typically very time-consuming. It requires many samples of $H(\mathbf{X})$ to obtain an accurate (small-variance) estimator $\widehat{\ell}(\mathbf{u})$ of $\ell(\mathbf{u})$. This, in turn, implies that one is required to solve the linear program many times.

To speed up the simulation process, we shall use a parametric importance sampling pdf $f(\mathbf{x}; \mathbf{v})$. We can use the CE program (5.65) to find a "good" reference vector $\mathbf{v}$, where we choose the original parameter vector $\mathbf{u}$ as our (trial) vector $\mathbf{w}$. After $\widehat{\mathbf{v}}$ is obtained, we iterate (5.65) once more with $\mathbf{w} = \widehat{\mathbf{v}}$ to obtain the final $\mathbf{v}$. Since the components of $\mathbf{X}$ are independent and belong to an exponential family parameterized by the mean, we can directly apply the explicit updating formulas (5.69) to solve (5.65).

5.9.2.1 Numerical Results Here we present numerical results for the CONFTRA model, which includes 32 generators connected by 38 lines. We took a sample size $N = 500$ for each CE iteration and a sample $N = 1000$ for the resulting importance sampling estimator.

Define the efficiency of the importance sampling estimator $\widehat{\ell}(\mathbf{u}; \mathbf{v})$ relative to the CMC one $\widehat{\ell}(\mathbf{u})$ as

$$\varepsilon = \frac{\mathrm{Var}_{\mathbf{u}}(\widehat{\ell}(\mathbf{u}))}{\mathrm{Var}_{\mathbf{v}}(\widehat{\ell}(\mathbf{u}; \mathbf{v}))} .$$

Table 5.8 represents the original values $\mathbf{u}_i$ and the reference values $\mathbf{v}_i$ obtained by using (5.65). The numbers 1–32 correspond to the generators and the numbers 33–70 correspond to the lines. Note that the data are presented only for those lines and generators for which v_i differs from u_i, $i = 1, \ldots, n$ by at least 0.001.

Table 5.8 The original parameters u_i and the reference parameters v_i obtained from (5.65).

i	u_i	v_i	i	u_i	v_i
1	0.1000	0.1091	22	0.1200	0.5909
3	0.0200	0.0303	23	0.1200	0.6570
5	0.1000	0.1061	26	0.0100	0.0119
6	0.1000	0.1064	27	0.0100	0.0134
7	0.0200	0.0369	28	0.0100	0.0208
8	0.0200	0.0267	30	0.0400	0.0828
9	0.0400	0.0603	31	0.0400	0.0954
10	0.0400	0.0814	32	0.0800	0.4241
12	0.0500	0.1462	34	0.0011	0.0015
13	0.0500	0.1461	40	0.0011	0.0025
14	0.1500	0.1405	42	0.0040	0.0057
15	0.0200	0.0244	49	0.0877	0.0925
18	0.0200	0.0233	51	0.0013	0.0248
20	0.0400	0.0773	53	0.0013	0.0019
21	0.0400	0.0680	60	0.0013	0.0030

Table 5.9 represents the point estimators $\widehat{\ell}(\mathbf{u})$ and $\widehat{\ell}(\mathbf{u};\mathbf{v})$, their associated sample variances, and the estimated efficiency ε of the importance sampling estimator $\widehat{\ell}(\mathbf{u};\mathbf{v})$ relative to the CMC one $\widehat{\ell}(\mathbf{u})$ as functions of the sample size N. Note that in our experiments the CMC estimator used all N replications, while the importance sampling estimator used only $N - N_1$ replications, since $N_1 = 1000$ samples were used to estimate the reference parameter $\mathbf{v}$.

Table 5.9 The efficiency ε of the importance sampling estimator $\widehat{\ell}(\mathbf{u};\mathbf{v})$ relative to the CMC one $\widehat{\ell}(\mathbf{u})$ as functions of the sample size N.

N	$\widehat{\ell}(\mathbf{u})$	$\widehat{\ell}(\mathbf{u};\mathbf{v})$	$\mathrm{Var}_{\mathbf{u}}(\widehat{\ell}(\mathbf{u}))$	$\mathrm{Var}_{\mathbf{v}}(\widehat{\ell}(\mathbf{u};\mathbf{v}))$	ε
2000	15.0260	14.4928	4.55	0.100	45.5
4000	14.6215	14.4651	1.09	0.052	21.0
6000	14.0757	14.4861	0.66	0.036	18.3
8000	14.4857	14.4893	0.53	0.027	19.6
10000	14.8674	14.4749	0.43	0.021	20.5
12000	14.7839	14.4762	0.35	0.017	20.6
14000	14.8053	14.4695	0.30	0.015	20.0
16000	15.0781	14.4657	0.28	0.013	21.5
18000	14.8278	14.4607	0.24	0.011	21.8
20000	14.8048	14.4613	0.22	0.010	22.0

From the data in Table 5.9, if follows that the importance sampling estimator $\widehat{\ell}(\mathbf{u};\mathbf{v})$ is more efficient than the CMC one by at least a factor of 18.

Table 5.8 indicates that only a few of the reference parameters v_i, namely those numbered 12, 13, 22, 23, and 32 out of a total of 70, called the *bottleneck parameters*, differ significantly from their corresponding original values u_i, $i = 1, \ldots, 70$. This implies that instead of solving the original 70-dimensional CE program (5.65) one could solve, in fact, only a 5-dimensional one. These bottleneck components could be efficiently identified by using the screening algorithm developed in [22]. Motivated by this screening algorithm, we solved the 5-dimensional CE program instead of the 70-dimensional one while keeping $v_i = u_i$ for the remaining 65 parameters. In this case, we obtained better results than those in Table 5.9; the resulting importance sampling estimator $\widehat{\ell}(\mathbf{u};\mathbf{v})$ was more efficient than the CMC one by at least a factor of 20. The reason for that is obvious; the 65 nonbottleneck parameters $v_i \neq u_i$ contributed to the importance sampling estimator (and, thus, to the data in Table 5.9) nothing but noise and instability via the likelihood ratio term W.

Note finally that we performed similar experiments with much larger electric power models. We found that the original importance sampling estimator $\widehat{\ell}(\mathbf{u};\mathbf{v})$ performs poorly for $n \geqslant 300$. Screening, however, improves the performance dramatically. In particular, we found that the efficiency of the importance sampling estimator $\widehat{\ell}(\mathbf{u};\mathbf{v})$ with screening depends mainly on the number of bottleneck parameters rather than on n. Our extensive numerical studies indicate that the importance sampling method still performs quite reliably if $n \leqslant 1000$, provided that the number of bottleneck parameters does not exceed 100.

PROBLEMS

5.1 Consider the integral $\ell = \int_a^b H(x)\,\mathrm{d}x = (b-a)\,\mathbb{E}[H(X)]$, with $X \sim \mathsf{U}(a,b)$. Let $X_1,\ldots,X_N$ be a random sample from $\mathsf{U}(a,b)$. Consider the estimators $\widehat{\ell} = \frac{1}{N}\sum_{i=1}^N H(X_i)$ and $\widehat{\ell}_1 = \frac{1}{2N}\sum_{i=1}^N \{H(X_i) + H(b+a-X_i)\}$. Prove that if $H(x)$ is monotonic in x, then

$$\mathrm{Var}(\widehat{\ell}_1) \leqslant \frac{1}{2}\mathrm{Var}(\widehat{\ell})\,.$$

In other words, using antithetic random variables is more accurate than using CMC.

5.2 Estimate the expected length of the shortest path for the bridge network in Example 5.1. Use both the CMC estimator (5.8) and the antithetic estimator (5.9). For both cases, take a sample size of N =100,000. Suppose that the lengths of the links $X_1,\ldots,X_5$ are exponentially distributed, with means $1, 1, 0.5, 2, 1.5$. Compare the results.

5.3 Use the batch means method to estimate the expected stationary waiting time in a $GI/G/1$ queue via Lindley's equation for the case where the interarrival times are $\mathsf{Exp}(1/2)$ distributed and the service times are $\mathsf{U}[0.5,2]$ distributed. Take a simulation run of M = 10,000 customers, discarding the first $K = 100$ observations. Examine to what extent variance reduction can be achieved by using antithetic random variables.

5.4 Run the stochastic shortest path problem in Example 5.4 and estimate the performance $\ell = \mathbb{E}[H(\mathbf{X})]$ from 1000 independent replications, using the given (C_1, C_2, C_3, C_4) as the vector of control variables, assuming that $X_i \sim \mathsf{Exp}(1)$, $i = 1,\ldots,5$. Compare the results with those obtained with the CMC method.

5.5 Estimate the expected waiting time of the fourth customer in a $GI/G/1$ queue for the case where the interarrival times are $\mathsf{Exp}(1/2)$ distributed and the service times are $\mathsf{U}[0.5,2]$ distributed. Use Lindley's equation and control variables, as described in Example 5.5. Generate $N = 1000$ replications of W_4 and provide a 95% confidence interval for $\mathbb{E}[W_4]$.

5.6 Prove that for any pair of random variables (U, V),

$$\mathrm{Var}(U) = \mathbb{E}[\,\mathrm{Var}(U \mid V)] + \mathrm{Var}(\,\mathbb{E}[U \mid V]\,)\,.$$

(Hint: Use the facts that $\mathbb{E}[U^2] = \mathbb{E}[\,\mathbb{E}[U^2 \mid V]\,]$ and $\mathrm{Var}(X) = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$.)

5.7 Let $R \sim \mathsf{G}(p)$ and define $S_R = \sum_{i=1}^R X_i$, where $X_1, X_2, \ldots$ is a sequence of iid $\mathsf{Exp}(\lambda)$ random variables that are independent of R.

- **a)** Show, that $S_R \sim \mathsf{Exp}(\lambda p)$. (Hint: the easiest way is to use transform methods and conditioning.)
- **b)** For $\lambda = 1$ and $p = 1/10$, estimate $\mathbb{P}(S_R > 10)$ using CMC with a sample size of $N = 1000$.
- **c)** Repeat b), now using the conditional Monte Carlo estimator (5.23). Compare the results with those of a) and b).

5.8 Consider the random sum S_R in Problem 5.7, with parameters $p = 0.25$ and $\lambda = 1$. Estimate $\mathbb{P}(S_R > 10)$ via stratification using strata corresponding to the partition of events $\{R = 1\}, \{R = 2\}, \ldots, \{R = 7\}$, and $\{R > 7\}$. Allocate a total of N = 10,000 samples via both $N_i = p_i N$ and the optimal N_i^* in (5.36), and compare the results. For the second method, use a simulation run of size 1000 to estimate the standard deviations $\{\sigma_i\}$.

5.9 Show that the solution to the minimization program

$$\min_{N_1,\ldots,N_m} \sum_{i=1}^{m} \frac{p_i^2 \sigma_i^2}{N_i} \quad \text{such that} \quad N_1 + \cdots + N_m = N \,,$$

is given by (5.36). This justifies the stratified sampling Theorem 5.5.1.

5.10 Use Algorithm 5.4.2 and (5.27) to estimate the reliability of the bridge reliability network in Example 4.1 on page 98 via permutation Monte Carlo. Consider two cases, where the link reliabilities are given by $\mathbf{p} = (0.3, 0.1, 0.8, 0.1, 0.2)$ and $\mathbf{p} = (0.95, 0.95, 0.95, 0.95, 0.95)$, respectively. Take a sample size of $N = 2000$.

5.11 Repeat Problem 5.10, using Algorithm 5.4.3. Compare the results.

5.12 This exercise discusses the counterpart of Algorithm 5.4.3 involving minimal paths rather than minimal cuts. A state vector $\mathbf{x}$ in the reliability model of Section 5.4.1 is called a *path vector* if $H(\mathbf{x}) = 1$. If in addition $H(\mathbf{y}) = 0$ for all $\mathbf{y} < \mathbf{x}$, then $\mathbf{x}$ is called the *minimal path vector*. The corresponding set $A = \{i : x_i = 1\}$ is called the *minimal path set*; that is, a minimal path set is a minimal set of components whose *functioning* ensures the functioning of the system. If $A_1, \ldots, A_m$ denote all the minimal paths sets, then the system is functioning if and only if all the components of at least one minimal path set are functioning.

a) Show that

$$H(\mathbf{x}) = \max_k \prod_{i \in A_k} x_i = 1 - \prod_{k=1}^{m} \left(1 - \prod_{i \in A_k} x_i\right). \tag{5.112}$$

b) Define

$$Y_k = \prod_{i \in A_k} X_i, \; k = 1, \ldots, m,$$

that is, Y_k is the indicator of the event that all components in A_i are functioning. Apply Proposition 5.4.1 to the sum $S = \sum_{k=1}^{m} Y_k$ and devise an algorithm similar to Algorithm 5.4.3 to estimate the reliability $r = \mathbb{P}(S > 0)$ of the system.

c) Test this algorithm on the bridge reliability network in Example 4.1.

5.13 Prove (see (5.45)) that the solution of

$$\min_g \mathrm{Var}_g \left(H(\mathbf{X}) \frac{f(\mathbf{X})}{g(\mathbf{X})} \right)$$

is

$$g^*(\mathbf{x}) = \frac{|H(\mathbf{x})| \, f(\mathbf{x})}{\int |H(\mathbf{x})| \, f(\mathbf{x}) \, \mathrm{d}\mathbf{x}} \,.$$

5.14 Let $Z \sim \mathsf{N}(0,1)$. Estimate $\mathbb{P}(Z > 4)$ via importance sampling, using the following shifted exponential sampling pdf:

$$g(x) = \mathrm{e}^{-(x-4)}, \quad x \geqslant 4 \,.$$

Choose N large enough to obtain accuracy to at least three significant digits and compare with the exact value.

5.15 Verify that the VM program (5.44) is equivalent to minimizing the Pearson χ^2 discrepancy measure (see Remark 1.14.1) between the zero-variance pdf g^* in (5.46) and the importance sampling density g. In this sense, the CE and VM methods are similar, because the CE method minimizes the Kullback–Leibler distance between g^* and g.

5.16 Repeat Problem 5.2 using importance sampling, where the lengths of the links are exponentially distributed with means $v_1, \ldots, v_5$. Write down the deterministic CE updating formulas and estimate these via a simulation run of size 1000 using $\mathbf{w} = \mathbf{u}$.

5.17 Consider the natural exponential family ((A.9) in the Appendix). Show that (5.62), with $\mathbf{u} = \boldsymbol{\theta}_0$ and $\mathbf{v} = \boldsymbol{\theta}$, reduces to solving

$$\mathbb{E}_{\boldsymbol{\theta}_0}\left[H(\mathbf{X})\left(\frac{\nabla c(\boldsymbol{\theta})}{c(\boldsymbol{\theta})} + \mathbf{t}(\mathbf{X})\right)\right] = \mathbf{0}\,. \tag{5.113}$$

5.18 As an application of (5.113), suppose that we wish to estimate the expectation of $H(X)$, with $X \sim \mathsf{Exp}(\lambda_0)$. Show that the corresponding CE optimal parameter is

$$\lambda^* = \frac{\mathbb{E}_{\lambda_0}[H(X)]}{\mathbb{E}_{\lambda_0}[H(X)X]}\,.$$

Compare with (A.15) in the Appendix. Explain how to estimate λ^* via simulation.

5.19 Let $X \sim \mathsf{Weib}(\alpha, \lambda_0)$. We wish to estimate $\ell = \mathbb{E}_{\lambda_0}[H(X)]$ via the SLR method, generating samples from $\mathsf{Weib}(\alpha, \lambda)$ — thus changing the scale parameter λ but keeping the scale parameter α fixed. Use (5.113) and Table A.1 in the Appendix to show that the CE optimal choice for λ is

$$\lambda^* = \left(\frac{\mathbb{E}_{\lambda_0}[H(X)]}{\mathbb{E}_{\lambda_0}[H(X)\,X^\alpha]}\right)^{1/\alpha}\,.$$

Explain how we can estimate λ^* via simulation.

5.20 Let $X_1, \ldots, X_n$ be independent $\mathsf{Exp}(1)$ distributed random variables. Let $\mathbf{X} = (X_1, \ldots, X_n)$ and $S(\mathbf{X}) = X_1 + \cdots + X_n$. We wish to estimate $\mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$ via importance sampling, using $X_i \sim \mathsf{Exp}(\theta)$, for all i. Show that the CE optimal parameter θ^* is given by

$$\theta^* = \frac{\mathbb{E}[I_{\{S(\mathbf{X})\geqslant\gamma\}}]}{\mathbb{E}[I_{\{S(\mathbf{X})\geqslant\gamma\}}\,\overline{X}]}\,,$$

with $\overline{X} = (X_1 + \cdots + X_n)/n$ and $\mathbb{E}$ indicating the expectation under the original distribution (where each $X_i \sim \mathsf{Exp}(1)$).

5.21 Consider Problem 5.19. Define $G(z) = z^{1/\alpha}/\lambda_0$ and $\widetilde{H}(z) = H(G(z))$.

a) Show that if $Z \sim \mathsf{Exp}(1)$, then $G(Z) \sim \mathsf{Weib}(\alpha, \lambda_0)$.

b) Explain how to estimate ℓ via the TLR method.

c) Show that the CE optimal parameter for Z is given by

$$\eta^* = \frac{\mathbb{E}_\eta[\widetilde{H}(Z)\,W(Z;1,\eta)]}{\mathbb{E}_\eta[\widetilde{H}(Z)\,Z\,W(Z;1,\eta)]}\,,$$

where $W(Z;1,\eta)$ is the ratio of the $\mathsf{Exp}(1)$ and $\mathsf{Exp}(\eta)$ pdfs.

5.22 Assume that the expected performance can be written as $\ell = \sum_{i=1}^{m} a_i \ell_i$, where $\ell_i = \int H_i(\mathbf{x})\, d\mathbf{x}$, and the a_i, $i = 1, \ldots, m$ are known coefficients. Let $Q(\mathbf{x}) = \sum_{i=1}^{m} a_i H_i(\mathbf{x})$. For any pdf g dominating $Q(\mathbf{x})$, the random variable

$$L = \sum_{i=1}^{m} a_i \frac{H_i(\mathbf{X})}{g(\mathbf{X})} = \frac{Q(\mathbf{X})}{g(\mathbf{X})} ,$$

where $\mathbf{X} \sim g$, is an unbiased estimator of ℓ — note that there is only one sample. Prove that L attains the smallest variance when $g = g^*$, with

$$g^*(\mathbf{x}) = |Q(\mathbf{x})| / \int |Q(\mathbf{x})|\, d\mathbf{x} ,$$

and that

$$\mathrm{Var}_{g^*}(L) = \left(\int |Q(\mathbf{x})|\, d\mathbf{x} \right)^2 - \ell^2 .$$

5.23 The Hit-or-Miss Method. Suppose that the sample performance function, H, is bounded on the interval $[0, b]$, say, $0 \leqslant H(x) \leqslant c$ for $x \in [0, b]$. Let $\ell = \int H(x)\, dx = b\, \mathbb{E}[H(X)]$, with $X \sim \mathsf{U}[0, b]$. Define an estimator of ℓ by

$$\widehat{\ell}^{\,h} = \frac{b\,c}{N} \sum_{i=1}^{N} I_{\{Y_i < H(X_i)\}} ,$$

where $\{(X_i, Y_i) : j = 1, \ldots, N\}$ is a sequence of points uniformly distributed over the rectangle $[0, b] \times [0, c]$ (see Figure 5.6). The estimator $\widehat{\ell}^{\,h}$ is called the *hit-or-miss estimator*, since a point (X, Y) is accepted or rejected depending on whether that point falls inside or outside the shaded area in Figure 5.6, respectively. Show that the hit-or-miss estimator has a larger variance than the CMC estimator,

$$\widehat{\ell} = \frac{b}{N} \sum_{i=1}^{N} H(X_i) ,$$

with $X_1, \ldots, X_N$ a random sample from $\mathsf{U}[0, b]$.

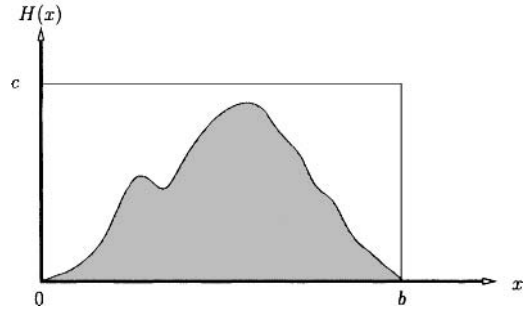


Figure 5.6 The hit-or-miss method.

Further Reading

The fundamental paper on variance reduction techniques is Kahn and Marshal [16]. There are a plenty of good Monte Carlo textbooks with chapters on variance reduction techniques. Among them are [10], [13], [17], [18], [20], [23], [24], [26], [27], and [34]. For a comprehensive study of variance reduction techniques see Fishman [10] and Rubinstein [28]. Asmussen and Glynn [2] provide a modern treatment of variance reduction and rare-event simulation.

An introduction to reliability models may be found in [12]. For more information on variance reduction in the presence of heavy-tailed distributions see also [1], [3], [4], and [7].

REFERENCES

1. S. Asmussen. Stationary distributions via first passage times. In J. H. Dshalalow, editor, *Advances in Queueing: Theory, Methods and Open Problems*, pages 79–102, New York, 1995. CRC Press.
2. S. Asmussen and P. W. Glynn. *Stochastic Simulation*. Springer-Verlag, New York, 2007.
3. S. Asmussen and D. P. Kroese. Improved algorithms for rare event simulation with heavy tails. *Advances in Applied Probability*, 38(2), 2006.
4. S. Asmussen, D. P. Kroese, and R. Y. Rubinstein. Heavy tails, importance sampling and cross-entropy. *Stochastic Models*, 21(1):57–76, 2005.
5. S. Asmussen and R. Y. Rubinstein. Complexity properties of steady-state rare-events simulation in queueing models. In J. H. Dshalalow, editor, *Advances in Queueing: Theory, Methods and Open Problems*, pages 429–462, New York, 1995. CRC Press.
6. W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, New York, 3rd edition, 1977.
7. P. T. de Boer, D. P. Kroese, and R. Y. Rubinstein. A fast cross-entropy method for estimating buffer overflows in queueing networks. *Management Science*, 50(7):883–895, 2004.
8. A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, New York, 2001.
9. T. Elperin, I. B. Gertsbakh, and M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40(5):572–581, 1991.
10. G. S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, New York, 1996.
11. S. Gal, R. Y. Rubinstein, and A. Ziv. On the optimality and efficiency of common random numbers. *Math. Comput. Simul.*, 26(6):502–512, 1984.
12. I. B. Gertsbakh. *Statistical Reliability Theory*. Marcel Dekker, New York, 1989.
13. P. Glasserman. *Monte Carlo Methods in Financial Engineering*. Springer-Verlag, New York, 2004.
14. D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 2nd edition, 1985.
15. S. Gunha, M. Pereira, and C. Oliveira L. Pinto. Composite generation and transmission reliability evaluation in large hydroelectric systems. *IEEE Transactions on Power Apparatus and Systems*, 104:2657–2663, 1985.
16. M. Kahn and A. W. Marshall. Methods of reducing sample size in Monte Carlo computations. *Operations Research*, 1:263–278, 1953.

17. J. P. C. Kleijnen. *Statistical Techniques in Simulation, Part 1.* Marcel Dekker, New York, 1974.

18. J. P. C. Kleijnen. Analysis of simulation with common random numbers: A note on Heikes et al. *Simuletter*, 11:7–13, 1976.

19. D. P. Kroese and R. Y. Rubinstein. The transform likelihood ratio method for rare event simulation with heavy tails. *Queueing Systems*, 46:317–351, 2004.

20. A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, New York, 3rd edition, 2000.

21. D. Lieber, A. Nemirovski, and R. Y. Rubinstein. A fast Monte Carlo method for evaluation of reliability indices. *IEEE Transactions on Reliability Systems*, 48(3):256–261, 1999.

22. D. Lieber, R. Y. Rubinstein, and D. Elmakis. Quick estimation of rare events in stochastic networks. *IEEE Transaction on Reliability*, 46:254–265, 1997.

23. J. S. Liu. *Monte Carlo Strategies in Scientifi c Computing.* Springer-Verlag, New York, 2001.

24. D. L. McLeish. *Monte Carlo Simulation and Finance.* John Wiley & Sons, New York, 2005.

25. M. F. Neuts. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach.* Dover Publications, New York, 1981.

26. C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer, New York, 2nd edition, 2004.

27. S. M. Ross. *Simulation.* Academic Press, New York, 3rd edition, 2002.

28. R. Y. Rubinstein. *Simulation and the Monte Carlo Method.* John Wiley & Sons, New York, 1981.

29. R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unifi ed Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning.* Springer-Verlag, New York, 2004.

30. R. Y. Rubinstein and R. Marcus. Efficiency of multivariate control variables in Monte Carlo simulation. *Operations Research*, 33:661–667, 1985.

31. R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling.* John Wiley & Sons, New York, 1998.

32. R. Y. Rubinstein and A. Shapiro. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization Via the Score Function Method.* John Wiley & Sons, New York, 1993.

33. R.Y. Rubinstein, M. Samorodnitsky, and M. Shaked. Antithetic variables, multivariate dependence and simulation of complex stochastic systems. *Management Science*, 31:66–77, 1985.

34. I. M. Sobol. *A Primer for the Monte Carlo Method.* CRC Press, Boca Raton, FL, 1994.

35. W. Whitt. Bivariate distributions with given marginals. *Annals of Statistics*, 4(6):1280–1289, 1976.

CHAPTER 6

MARKOV CHAIN MONTE CARLO

6.1 INTRODUCTION

In this chapter we present a powerful generic method, called *Markov chain Monte Carlo* (MCMC), for *approximately* generating samples from an arbitrary distribution. This, as we learned in Section 2.5, is typically not an easy task, in particular when $\mathbf{X}$ is a random vector with dependent components. An added advantage of MCMC is that it only requires specification of the target pdf up to a (normalization) constant.

The MCMC method is due to Metropolis et al. [17]. They were motivated by computational problems in statistical physics, and their approach uses the idea of generating a Markov chain whose limiting distribution is equal to the desired target distribution. There are many modifications and enhancement of the original Metropolis [17] algorithm, most notably the one by Hastings [10]. Nowadays, any approach that produces an ergodic Markov chain whose stationary distribution is the target distribution is referred to as MCMC or *Markov chain sampling* [19]. The most prominent MCMC algorithms are the Metropolis–Hastings and the Gibbs samplers, the latter being particularly useful in Bayesian analysis. Finally, MCMC sampling is the main ingredient in the popular *simulated annealing* technique [1] for discrete and continuous optimization.

The rest of this chapter is organized as follows. In Section 6.2 we present the classic Metropolis–Hastings algorithm, which simulates a Markov chain such that its stationary distribution coincides with the target distribution. An important special case is the *hit-and-run* sampler, discussed in Section 6.3. Section 6.4 deals with the *Gibbs sampler*, where the underlying Markov chain is constructed based on a sequence of conditional distributions.

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

Section 6.5 explains how to sample from distributions arising in the Ising and Potts models, which are extensively used in statistical mechanics, while Section 6.6 deals with applications of MCMC in Bayesian statistics. In Section 6.7 we show that both the Metropolis–Hastings and Gibbs samplers can be viewed as special cases of a general MCMC algorithm and present the slice and reversible jump samplers. Section 6.8 deals with the classic simulated annealing method for finding the global minimum of a multiextremal function, which is based on the MCMC method. Finally, Section 6.9 presents the perfect sampling method, for sampling exactly from a target distribution rather than approximately.

6.2 THE METROPOLIS–HASTINGS ALGORITHM

The main idea behind the Metropolis–Hastings algorithm is to simulate a Markov chain such that the stationary distribution of this chain coincides with the target distribution.

To motivate the MCMC method, assume that we want to generate a random variable X taking values in $\mathscr{X} = \{1, \ldots, m\}$, according to a target distribution $\{\pi_i\}$, with

$$\pi_i = \frac{b_i}{C}, \quad i \in \mathscr{X}, \tag{6.1}$$

where it is assumed that all $\{b_i\}$ are strictly positive, m is large, and the normalization constant $C = \sum_{i=1}^{m} b_i$ is difficult to calculate. Following Metropolis et al. [17], we construct a Markov chain $\{X_t,\ t = 0, 1, \ldots\}$ on $\mathscr{X}$ whose evolution relies on an arbitrary transition matrix $\mathbf{Q} = (q_{ij})$ in the following way:

- When $X_t = i$, generate a random variable Y satisfying $\mathbb{P}(Y = j) = q_{ij},\ j \in \mathscr{X}$. Thus, Y is generated from the m-point distribution given by the i-th row of $\mathbf{Q}$.
- If $Y = j$, let

$$X_{t+1} = \begin{cases} j & \text{with probability } \alpha_{ij} = \min\left\{\frac{\pi_j\, q_{ji}}{\pi_i\, q_{ij}}, 1\right\} = \min\left\{\frac{b_j\, q_{ji}}{b_i\, q_{ij}}, 1\right\}, \\ i & \text{with probability } 1 - \alpha_{ij}. \end{cases}$$

It follows that $\{X_t, t = 0, 1, \ldots\}$ has a one-step transition matrix $\mathbf{P} = (p_{ij})$ given by

$$p_{ij} = \begin{cases} q_{ij}\, \alpha_{ij}, & \text{if } i \neq j \\ 1 - \sum_{k \neq i} q_{ik}\, \alpha_{ik}, & \text{if } i = j\ . \end{cases} \tag{6.2}$$

Now it is easy to check (see Problem 6.1) that, with α_{ij} as above,

$$\pi_i\, p_{ij} = \pi_j\, p_{ji}, \quad i, j \in \mathscr{X}\ . \tag{6.3}$$

In other words, the *detailed balance equations* (1.38) hold, and hence the Markov chain is time reversible and has stationary probabilities $\{\pi_i\}$. Moreover, this stationary distribution is also the *limiting* distribution if the Markov chain is irreducible and aperiodic. Note that there is no need for the normalization constant C in (6.1) to define the Markov chain.

The extension of the above MCMC approach for generating samples from an arbitrary multidimensional pdf $f(\mathbf{x})$ (instead of π_i) is straightforward. In this case, the nonnegative probability transition function $q(\mathbf{x}, \mathbf{y})$ (taking the place of q_{ij} above) is often called the *proposal* or *instrumental* function. Viewing this function as a conditional pdf, one also writes

$q(\mathbf{y} \mid \mathbf{x})$ instead of $q(\mathbf{x}, \mathbf{y})$. The probability $\alpha(\mathbf{x}, \mathbf{y})$ is called the *acceptance probability*. The original Metropolis algorithm [17] was suggested for symmetric proposal functions, that is, for $q(\mathbf{x}, \mathbf{y}) = q(\mathbf{y}, \mathbf{x})$. Hastings modified the original MCMC algorithm to allow nonsymmetric proposal functions. Such an algorithm is called a *Metropolis–Hastings algorithm*. We call the corresponding Markov chain the *Metropolis–Hastings Markov chain*.

In summary, the Metropolis–Hastings algorithm, which, like the acceptance-rejection method, is based on a trial-and-error strategy, is comprised of the following iterative steps:

Algorithm 6.2.1 (Metropolis–Hastings Algorithm)

Given the current state $\mathbf{X}_t$*:*

1. *Generate* $\mathbf{Y} \sim q(\mathbf{X}_t, \mathbf{y})$.
2. *Generate* $U \sim \mathsf{U}(0,1)$ *and deliver*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y}, & \text{if } U \leq \alpha(\mathbf{X}_t, \mathbf{Y}) \\ \mathbf{X}_t & \text{otherwise}, \end{cases} \tag{6.4}$$

where

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\{\varrho(\mathbf{x}, \mathbf{y}), 1\}\ , \tag{6.5}$$

with

$$\varrho(\mathbf{x}, \mathbf{y}) = \frac{f(\mathbf{y})\, q(\mathbf{y}, \mathbf{x})}{f(\mathbf{x})\, q(\mathbf{x}, \mathbf{y})}\ . \tag{6.6}$$

By repeating Steps 1 and 2, we obtain a sequence $\mathbf{X}_1, \mathbf{X}_2, \ldots$ of *dependent* random variables, with $\mathbf{X}_t$ approximately distributed according to $f(\mathbf{x})$, for large t.

Since Algorithm 6.2.1 is of the acceptance-rejection type, its efficiency depends on the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$. Ideally, one would like $q(\mathbf{x}, \mathbf{y})$ to reproduce the desired pdf $f(\mathbf{y})$ as faithfully as possible. This clearly implies maximization of $\alpha(\mathbf{x}, \mathbf{y})$. A common approach [19] is to first parameterize $q(\mathbf{x}, \mathbf{y})$ as $q(\mathbf{x}, \mathbf{y}; \theta)$ and then use stochastic optimization methods to maximize this with respect to θ. Below we consider several particular choices of $q(\mathbf{x}, \mathbf{y})$.

■ **EXAMPLE 6.1 Independence Sampler**

The simplest Metropolis-type MCMC algorithm is obtained by choosing the proposal function $q(\mathbf{x}, \mathbf{y})$ to be independent of $\mathbf{x}$, that is, $q(\mathbf{x}, \mathbf{y}) = g(\mathbf{y})$ for some pdf $g(\mathbf{y})$. Thus, starting from a previous state $\mathbf{X}$ a candidate state $\mathbf{Y}$ is generated from $g(\mathbf{y})$ and accepted with probability

$$\alpha(\mathbf{X}, \mathbf{Y}) = \min\left\{\frac{f(\mathbf{Y})\, g(\mathbf{X})}{f(\mathbf{X})\, g(\mathbf{Y})}, 1\right\}.$$

This procedure is very similar to the original acceptance-rejection methods of Chapter 2 and, as in that method, it is important that the proposal distribution g is close to the target f. Note, however, that in contrast to the acceptance-rejection method the independence sampler produces *dependent* samples.

■ **EXAMPLE 6.2 Uniform Sampling**

Being able to sample *uniformly* from some discrete set $\mathscr{Y}$ is very important in many applications; see, for example, the algorithms for counting in Chapter 9. A simple general procedure is as follows. Define a *neighborhood* structure on $\mathscr{Y}$. Any neighborhood structure is allowed, as long as the resulting Metropolis–Hastings Markov chain is irreducible and aperiodic. Let $n_{\mathbf{x}}$ be the number of neighbors of a state $\mathbf{x}$. For the proposal distribution, we simply choose each possible neighbor of the current state $\mathbf{x}$ with equal probability. That is, $q(\mathbf{x}, \mathbf{y}) = 1/n_{\mathbf{x}}$. Since the target pdf $f(\mathbf{x})$ here is constant, the acceptance probability is

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\{n_{\mathbf{x}}/n_{\mathbf{y}}, 1\} .$$

By construction, the limiting distribution of the Metropolis-Hastings Markov chain is the uniform distribution on $\mathscr{Y}$.

■ **EXAMPLE 6.3 Random Walk Sampler**

In the random walk sampler the proposal state $\mathbf{Y}$, for a given current state $\mathbf{x}$, is given by $\mathbf{Y} = \mathbf{x} + \mathbf{Z}$, where $\mathbf{Z}$ is typically generated from some spherically symmetrical distribution (in the continuous case), such as $\mathsf{N}(\mathbf{0}, \Sigma)$. Note that the proposal function is symmetrical in this case; thus,

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\left\{\frac{f(\mathbf{y})}{f(\mathbf{x})}, 1\right\}. \tag{6.7}$$

■ **EXAMPLE 6.4**

Let the random vector $\mathbf{X} = (X_1, X_2)$ have the following two-dimensional pdf:

$$f(\mathbf{x}) = c \exp(-(x_1^2 x_2^2 + x_1^2 + x_2^2 - 8x_1 - 8x_2)/2) , \tag{6.8}$$

where $c \approx 1/20216.335877$ is a normalization constant. The graph of this density is depicted in Figure 6.1.

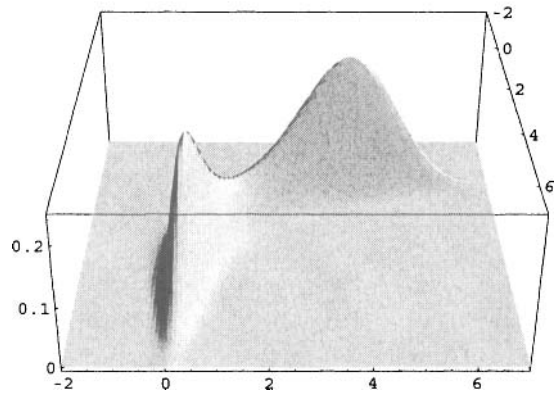


Figure 6.1 The density $f(x_1, x_2)$.

Suppose we wish to estimate $\ell = \mathbb{E}[X_1]$ via the CMC estimator

$$\widehat{\ell} = \frac{1}{N} \sum_{t=1}^{N} X_{t1},$$

using the random walk sampler to generate a dependent sample $\{\mathbf{X}_t\}$ from $f(\mathbf{x})$. A simple choice for the increment $\mathbf{Z}$ is to draw the components of $\mathbf{Z}$ independently, from a $\mathsf{N}(0, a^2)$ distribution for some $a > 0$. Note that if a is chosen too small, say less than 0.5, the components of the samples will be strongly positively correlated, which will lead to a large variance for $\widehat{\ell}$. On the other hand, for a too large, say greater than 10, most of the samples will be rejected, leading again to low efficiency. Below we choose a moderate value of a, say $a = 2$. The random walk sampler is now summarized as follows:

Procedure (Random Walk Sampler)

1. Initialize $\mathbf{X}_1 = (X_{11}, X_{12})$. Set $t = 1$.
2. Draw $Z_1, Z_2 \sim \mathsf{N}(0, 1)$ independently. Let $\mathbf{Z} = (Z_1, Z_2)$ and $\mathbf{Y} = \mathbf{X}_t + 2\,\mathbf{Z}$. Calculate $\alpha = \alpha(\mathbf{X}_t, \mathbf{Y})$ as in (6.7).
3. Draw $U \sim \mathsf{U}[0, 1]$. If $U < \alpha$, let $\mathbf{X}_{t+1} = \mathbf{Y}$; otherwise, let $\mathbf{X}_{t+1} = \mathbf{X}_t$.
4. Increase t by 1. If $t = N$ (sample size) stop; otherwise, repeat from Step 2.

We ran the above algorithm to produce $N = 10^5$ samples. The last few hundred of these are displayed in the left plot of Figure 6.2. We see that the samples closely follow the contour plot of the pdf, indicating that the correct region has been sampled. This is corroborated by the right plot of Figure 6.2, where we see that the histogram of the x_1 values is close to the true pdf (solid line).

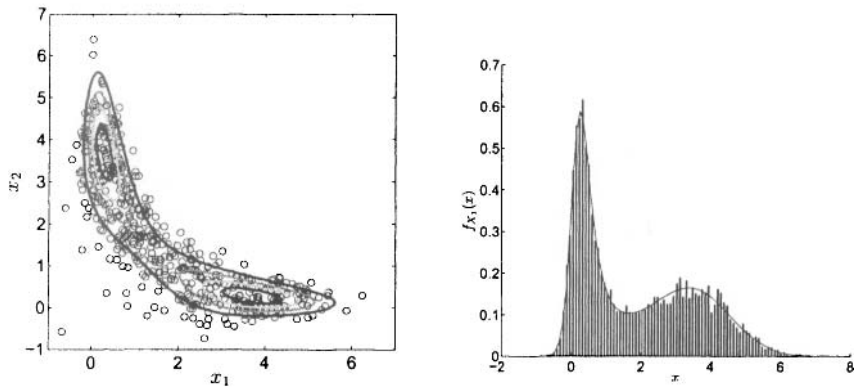


Figure 6.2 The left plot shows some samples of the random walk sampler along with several contour lines of f. The right plot shows the histogram of the x_1 values along with the true density of X_1.

We obtained an estimate $\widehat{\ell} = 1.89$ (the true value is $\mathbb{E}[X_1] \approx 1.85997$). To obtain a CI, we can use (4.18), where $\tilde{S}$ estimates the asymptotic variance, or employ the batch

means method of Section 4.3.2.1. Figure 6.3 displays the estimated (auto)covariance function $\widehat{R}(k)$ for $k = 0, 1, \ldots, 400$. We see that up to about 100 the covariances are nonnegligible. Thus, to estimate the variance of $\widehat{\ell}$, we need to include all nonzero terms in (4.17), not only the variance $R(0)$ of X_1. Summing over the first 400 lags, we obtained an estimate of 10.41 for the asymptotic variance. This gives an estimated relative error for $\widehat{\ell}$ of 0.0185 and an 95% CI of (1.82, 1.96). A similar CI was found when using the batch means method with 500 batches of size 200.

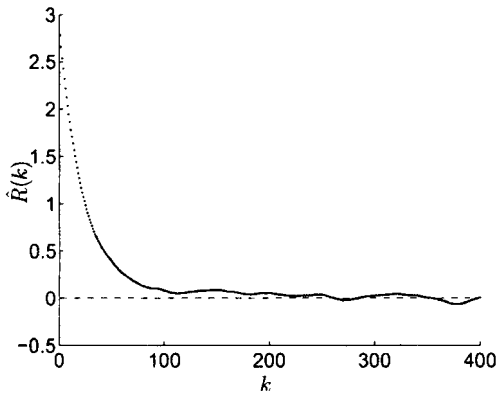


Figure 6.3 The estimated covariance function for the $\{X_{t1}\}$ for lags k up to 400.

While MCMC is a generic method and can be used to generate random samples virtually from any target distribution, regardless of its dimensionality and complexity, potential problems with the MCMC method are:

1. The resulting samples are often highly correlated.

2. Typically, it takes a considerable amount of time until the underlying Markov chain settles down to its steady state.

3. The estimates obtained via MCMC samples often tend to have much greater variances then those obtained from independent sampling of the target distribution. Various attempts have been made to overcome this difficulty. For details see, for example, [13] and [19].

Remark 6.2.1 At this point we must stress that although it is common practice to use MCMC to sample from $f(\mathbf{x})$ in order to estimate any expectation $\ell = \mathbb{E}_f[H(\mathbf{X})]$, the *actual* target for estimating ℓ is $g^*(\mathbf{x}) \propto |H(\mathbf{x})| f(\mathbf{x})$. Namely, sampling from $g^*(\mathbf{x})$ gives a minimum variance estimator (zero variance in the case $H(\mathbf{x}) \geqslant 0$). Thus, it is important to distinguish clearly between using MCMC for generating from some difficult pdf $f(\mathbf{x})$ and using MCMC to estimate a quantity such as ℓ. For the latter problem, much more efficient techniques can be used, such as importance sampling; moreover, a good importance sampling pdf can be obtained adaptively, as with the CE and TLR methods.

6.3 THE HIT-AND-RUN SAMPLER

The *hit-and-run* sampler, pioneered by Robert Smith [24], is among the first MCMC samplers in the category of *line samplers* [2]. As in the previous section, the objective is to sample from a target distribution $f(\mathbf{x})$ on $\mathscr{X} \subset \mathbb{R}^n$. Line samplers afford the opportunity to reach across the entire feasible region $\mathscr{X}$ in one step.

We first describe the original hit-and-run sampler for generating from a *uniform* distribution on a bounded open region $\mathscr{X}$ of $\mathbb{R}^n$. At each iteration, starting from a current point $\mathbf{x}$, a *direction vector* $\mathbf{d}$ is generated uniformly on the surface of an n-dimensional hypersphere. The intersection of the corresponding bidirectional line (through $\mathbf{x}$) and the enclosing box of $\mathscr{X}$ defines a line segment $\mathscr{L}$. The next point $\mathbf{y}$ is then selected uniformly from the intersection of $\mathscr{L}$ and $\mathscr{X}$.

Figure 6.4 illustrates the hit-and-run algorithm for generating uniformly from the set $\mathscr{X}$ (the gray region), which is bounded by a square. Given the point $\mathbf{x}$ in $\mathscr{X}$, a random direction $\mathbf{d}$ is generated, which defines the line segment $\mathscr{L} = uv$. Then a point $\mathbf{y}$ is chosen uniformly on $\mathscr{M} = \mathscr{L} \cap \mathscr{X}$, for example, by the acceptance-rejection method; that is, one generates a point uniformly on $\mathscr{L}$ and then accepts this point only if it lies in $\mathscr{X}$.

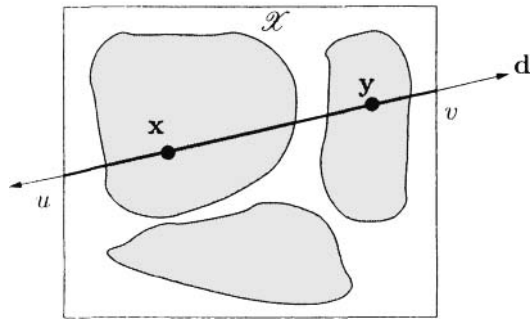


Figure 6.4 Illustration of the hit-and-run algorithm on a square in two dimensions.

Smith [24] showed that hit-and-run asymptotically generates uniformly distributed points over *arbitrary* open regions of $\mathbb{R}^n$. One desirable property of hit-and-run is that it can globally reach any point in the set in one step, that is, there is a strictly positive probability of sampling any neighborhood in the set. This property, coupled with a symmetry property, is important in deriving the limiting distribution. Lovász [14] proved that hit-and-run on a convex body in n dimensions produces an approximately uniformly distributed sample point in polynomial time, $\mathcal{O}(n^3)$, the best-known bound for such a sampling algorithm. He noted that the hit-and-run algorithm appears in practice to offer the most rapid convergence to a uniform distribution [14, 15]. Hit-and-run is unique in that it only takes polynomial time to get out of a corner; in contrast, *ball walk* takes exponential time to get out of a corner [16].

Note that the hit-and-run algorithm described above is a special case of the Metropolis–Hastings Algorithm 6.2.1, where the proposal function $q(\mathbf{x}, \mathbf{y})$ is symmetric and the target $f(\mathbf{x})$ is constant. It follows that each candidate point is accepted with probability 1. To generate from a *general* strictly positive continuous pdf $f(\mathbf{x})$, one can simply modify the

above uniform hit-and-run algorithm by accepting the candidate $\mathbf{y}$ with probability

$$\alpha(\mathbf{x}, \mathbf{y}) = \min\{f(\mathbf{y})/f(\mathbf{x}), 1\}\ , \tag{6.9}$$

as in (6.4) (note that $q(\mathbf{y}, \mathbf{x})/q(\mathbf{x}, \mathbf{y})$ equals 1). Thus, the general hit-and-run algorithm with the above Metropolis acceptance criterion is summarized as follows [20].

Algorithm 6.3.1 (Continuous Hit-and-Run Algorithm)

1. *Initialize* $\mathbf{X}_1 \in \mathscr{X}$ *and set* $t = 1$.
2. *Generate a random direction* $\mathbf{d}_t$ *according to a uniform distribution on the unit* n*-dimensional hypersphere.*
3. *Generate a candidate point* $\mathbf{Y} = \mathbf{X}_t + \lambda \mathbf{d}_t$ *uniformly distributed over the line set*

$$\mathscr{M}_t = \{\mathbf{x} : \mathbf{x} \in \mathscr{X} \text{ and } \mathbf{x} = \mathbf{X}_t + \lambda \mathbf{d}_t,\ \lambda \in \mathbb{R}\}\ .$$

   If $\mathscr{M}_t = \emptyset$, *go to Step 2.*

4. *Set*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y} & \text{with probability } \alpha(\mathbf{X}_t, \mathbf{Y}) \text{ in (6.9)} \\ \mathbf{X}_t & \text{otherwise.} \end{cases}$$

5. *If a stopping criterion is met, stop. Otherwise increment* t *and return to Step 2.*

Hit-and-run has been very successful over continuous domains. Recently an analogous sampler over a discrete domain has been developed in Baumert et al. [4]. Discrete hit-and-run generates two independent random walks on a lattice to form a bidirectional path that is analogous to the random bidirectional line generated in continuous hit-and-run. It then randomly selects a (feasible) discrete point along that path as the candidate point. By adding a Metropolis acceptance-rejection step, discrete hit-and-run converges to an arbitrary discrete target pdf f.

Let $\mathscr{X}$ be a bounded subset of $\mathbb{Z}^n$ — the set of n-dimensional vectors with integer valued coordinates — that is contained in the hyperrectangle $\mathscr{R} = \{\mathbf{x} \in \mathbb{Z}^n : l_i \le x_i \le u_i,\ i = 1, \ldots, n\}$. The discrete hit-and-run algorithm is stated below.

Algorithm 6.3.2 (Discrete Hit-and-Run Algorithm)

1. *Initialize* $\mathbf{X}_1 \in \mathscr{X}$ *and set* $t = 1$.
2. *Generate a bidirectional walk by generating two independent nearest neighbor random walks in* $\mathscr{R}$ *that start at* $\mathbf{X}_t$ *and end when they step out of* $\mathscr{R}$. *One random walk is called the* forward walk *and the other is called the* backward walk. *The bidirectional walk may have loops but has finite length with probability 1. The sequence of points visited by the bidirectional walk is stored in an ordered list, denoted* $\mathscr{L}_t$.
3. *Generate a candidate point* $\mathbf{Y}$ *uniformly distributed on the intersection* $\mathscr{M}_t = \mathscr{X} \cap \mathscr{L}_t$.
4. *Set*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Y} & \text{with probability } \alpha(\mathbf{X}_t, \mathbf{Y}) \text{ in (6.9)} \\ \mathbf{X}_t & \text{otherwise.} \end{cases}$$

5. *If a stopping criterion is met, stop. Otherwise, increment* t *and return to Step 2.*

Figure 6.5 illustrates the discrete hit-and-run algorithm with a bidirectional walk starting at point $\mathbf{x}$ on the discrete points in a square. The feasible set $\mathscr{X}$ is indicated by the three disconnected shaded regions. The candidate point $\mathbf{y}$ is chosen uniformly from the points in the bidirectional walk that are also in $\mathscr{X}$. It is accepted according to the Metropolis acceptance probability,

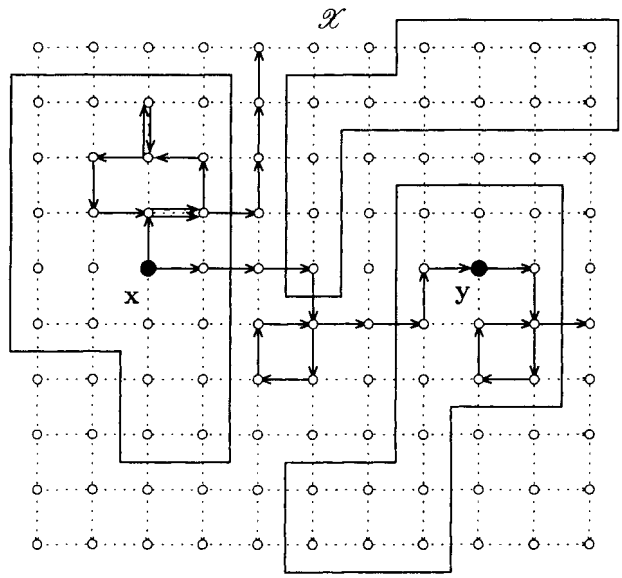


Figure 6.5 Illustration of the bidirectional walk in two dimensions.

It is shown in [4] that in several situations the rate of convergence of the discrete hit-and-run algorithm is polynomial of order n^4. Convergence to the target distribution has been proved in [4] for several variations of the bidirectional walk. One such candidate point generator, called *sphere biwalk*, not only converges to the target discrete pdf but also converges to continuous hit-and-run as the grid of the finite domain becomes finer [22].

The hit-and-run algorithm can be embedded within an optimization framework to yield two global optimization algorithms: *hide-and-seek* [20] and *improving hit-and-run* [26]. The latter has been applied successfully to practical problems including composite material design and shape optimization and has been shown to have polynomial complexity, on average, for a class of quadratic programs. In Section 6.8 we show how to turn an MCMC sampler into an optimization algorithm using simulated annealing.

6.4 THE GIBBS SAMPLER

The *Gibbs sampler* (Geman and Geman [6]) uses a somewhat different methodology from the Metropolis–Hastings algorithm and is particularly useful for generating n-dimensional random vectors. The distinguishing feature of the Gibbs sampler is that the underlying Markov chain is constructed, in a deterministic or random fashion, from a sequence of conditional distributions.

Gibbs sampling is advantageous if it is easier to sample from the conditional distributions than from the joint distribution. The essential idea of the Gibbs sampler — updating one part of the previous element while keeping the other parts fixed — is useful in many instances where the state variable is a random variable taking values in a general space, not just in $\mathbb{R}^n$; see [11].

Suppose that we wish to sample a random vector $\mathbf{X} = (X_1, \ldots, X_n)$ according to a target pdf $f(\mathbf{x})$. Let $f(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ represent the conditional pdf of the i-th component, X_i, given the other components $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. The Gibbs sampler consists of the following iterative steps.

Algorithm 6.4.1 (Gibbs Sampler)

A. For a given $\mathbf{X}_t$, *generate* $\mathbf{Y} = (Y_1, \ldots, Y_n)$ *as follows:*

1. *Draw* Y_1 *from the conditional pdf* $f(x_1 \mid X_{t,2}, \ldots, X_{t,n})$.
2. *Draw* Y_i *from* $f(x_i \mid Y_1, \ldots, Y_{i-1}, X_{t,i+1}, \ldots, X_{t,n})$, $\quad i = 2, \ldots, n-1$.
3. *Draw* Y_n *from* $f(x_n \mid Y_1, \ldots, Y_{n-1})$.

B. Let $\mathbf{X}_{t+1} = \mathbf{Y}$.

Note that in the Gibbs sampler *all* samples are accepted, in contrast to the Metropolis–Hastings algorithm. We will see in Section 6.7 that under mild conditions the limiting distribution of the process $\{\mathbf{X}_t, t = 1, 2, \ldots\}$, generated via the Gibbs sampler, is precisely $f(\mathbf{x})$. Moreover, under some other simple conditions, it can be shown (see [13], [19]) that the convergence to the desired pdf is geometrically fast.

■ **EXAMPLE 6.5 Example 6.4 (Continued)**

We shall show how to sample easily from the pdf f in (6.8) via the Gibbs sampler. Namely, writing

$$f(x, y) = c_1(y) \exp\left(-\frac{1+y^2}{2}\left(x - \frac{4}{1+y^2}\right)^2\right),$$

where $c_1(y)$ depends only on y, we see that, conditional on y, X has a normal distribution with expectation $4/(1+y^2)$ and variance $1/(1+y^2)$. The conditional distribution of Y given x follows in the same way. The corresponding Gibbs sampler is thus:

Procedure

1. Initialize X_1 and Y_1. Set $t = 1$.
2. If t is odd, draw $Z \sim \mathsf{N}(0, 1)$. Put $a = 1/(1 + Y_t^2)$. Set $Y_{t+1} = 4a + Z\sqrt{a}$ and $X_{t+1} = X_t$.
3. If t is even, draw $Z \sim \mathsf{N}(0, 1)$. Put $a = 1/(1 + X_t^2)$. Set $X_{t+1} = 4a + Z\sqrt{a}$ and $Y_{t+1} = Y_t$.
4. Increase t by 1. If $t = N$ (sample size) stop; otherwise, repeat from Step 2.

Remark 6.4.1 (Systematic and Random Gibbs Samplers) Note that Algorithm 6.4.1 presents a *systematic* coordinatewise Gibbs sampler. That is, the vector $\mathbf{X}$ is updated in a deterministic order: $1, 2, \ldots, n, 1, 2, \ldots$. In the *random* coordinatewise Gibbs sampler the coordinates are chosen randomly, such as by generating them independently from a discrete uniform n-point pdf. In that case the Gibbs sampler can be viewed as an instance of the Metropolis–Hastings sampler, namely, with the transition function

$$q(\mathbf{x}, \mathbf{y}) = \frac{1}{n} f(y_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \frac{1}{n} \frac{f(\mathbf{y})}{\sum_{y_i} f(\mathbf{y})} ,$$

where $\mathbf{y} = (x_1, \ldots, x_{i-1}, y_i, x_{i+1}, \ldots, x_n)$. Since $\sum_{y_i} f(\mathbf{y})$ can also be written as $\sum_{x_i} f(\mathbf{x})$, we have

$$\varrho(\mathbf{x}, \mathbf{y}) = \frac{f(\mathbf{y})\, q(\mathbf{y}, \mathbf{x})}{f(\mathbf{x})\, q(\mathbf{x}, \mathbf{y})} = \frac{f(\mathbf{y})\, f(\mathbf{x})}{f(\mathbf{x})\, f(\mathbf{y})} = 1 ,$$

so that the acceptance probability $\alpha(\mathbf{x}, \mathbf{y})$ is 1 in this case.

Here is another example of an application of the Gibbs sampler.

■ **EXAMPLE 6.6 Closed Network of Queues in a Product Form**

Consider m customers moving among n queues in a closed queueing network. Denote by $X_i(t)$ the number of customers in queue i, $i = 1, \ldots, n$, and let $\mathbf{X}(t) = (X_1(t), \ldots, X_n(t))$ and $\mathbf{x} = (x_1, \ldots, x_n)$. It is well known [21] that if the limit

$$\lim_{t \to \infty} \mathbb{P}(\mathbf{X}(t) = \mathbf{x}) = \pi(\mathbf{x})$$

exists, then, for exponentially distributed service times, the joint discrete pdf $\pi(\mathbf{x})$ can be written in *product form* as

$$\pi(\mathbf{x}) = C \prod_{i=1}^{n} f_i(x_i), \quad \text{for } \sum_{i=1}^{n} x_i = m , \tag{6.10}$$

where the $\{f_i(x_i),\ x_i \geq 0\}$ are *known* discrete pdfs, and C is a normalization constant. For a concrete example see Problem 6.11.

The constant C is in general difficult to compute. To proceed, writing $S(\mathbf{x}) = \sum_{i=1}^{n} x_i$ and $\mathscr{X}^* = \{\mathbf{x} : S(\mathbf{x}) = m\}$, we have

$$C^{-1} = \sum_{\mathbf{x} \in \mathscr{X}^*} \prod_{i=1}^{n} f_i(x_i) , \tag{6.11}$$

which requires the evaluation of the product of n pdfs for each $\mathbf{x}$ in the set $\mathscr{X}^*$. This set has a total of $|\mathscr{X}^*| = \binom{m+n-1}{n-1}$ elements (see Problem 6.10), which rapidly grows very large.

We now show how to compute C based on Gibbs sampling. To apply the Gibbs sampler, we need to be able to generate samples from the conditional distribution of X_i given the other components. Note that we only have to generate $X_1, \ldots, X_{n-1}$, since $X_n = m - \sum_{k=1}^{n-1} X_k$. For $i = 1, \ldots, n-1$ we have

$$f(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{n-1}) \propto f_i(x_i)\, f_n(m - \sum_{k=1}^{n-1} x_i) \tag{6.12}$$

for $x_i \in \{0, 1, \ldots, m - x_1 - \cdots - x_{i-1} - x_{i+1} - \cdots - x_{n-1}\}$. Sampling from these conditional pdfs can often be done efficiently, in particular when the $\{f_i\}$ are members of an exponential family; see also Problem 6.11.

Now that we can sample (approximately) from $\pi(\mathbf{x})$, it is straightforward to estimate the normalization constant C by observing that

$$\mathbb{E}_\pi \left[\frac{1}{\prod_{i=1}^n f_i(X_i)} \right] = \sum_{\mathbf{x} \in \mathscr{X}^*} \frac{1}{\prod_{i=1}^n f_i(x_i)} C \prod_{i=1}^n f_i(x_i) = |\mathscr{X}^*| \, C \, .$$

This suggests the following estimator for C, obtained from a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from π:

$$\widehat{C} = \binom{m+n-1}{n-1}^{-1} \frac{1}{N} \sum_{k=1}^N \prod_{i=1}^n \frac{1}{f_i(X_{ki})} \, ,$$

where X_{ki} is the i-th component of $\mathbf{X}_k$.

6.5 ISING AND POTTS MODELS

6.5.1 Ising Model

The Ising model is one of the most popular and most extensively studied models in statistical mechanics. It describes the interaction of idealized magnets, called *spins*, that are located on a two- or three-dimensional lattice. In the basic two-dimensional case the spins are located on the lattice $\{1, \ldots, n\} \times \{1, \ldots, n\}$, and each of the n^2 sites has four nearest neighbors, possibly including boundary sites, which "wrap around" to the other side of the lattice, creating a so-called *torus*. See Figure 6.6, where the four light gray sites are the neighbors of the dark gray site.

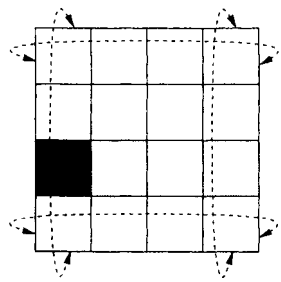


Figure 6.6 The boundary sites wrap around. The neighbors of the dark gray site are the light gray sites.

Let $\{1, \ldots, n^2\}$ be an enumeration of the sites. Each spin can be in one of two states: -1 or 1. Each of the 2^{n^2} *configurations* of spins $\mathbf{s} = (s_1, \ldots, s_{n^2})$ carries an amount of *total energy*

$$E(\mathbf{s}) = -A \sum_{i \leftrightarrow j} s_i \, s_j - B \sum_i s_i \, ,$$

where A and B are constants; in most studies $A = 1$ and $B = 0$, which we will now assume. The quantities $\sum_{i \leftrightarrow j} s_i \, s_j$ and $\sum_i s_i$ are called the *interaction energy* and *magnetization*,

respectively. The notation $\sum_{i \leftrightarrow j}$ indicates that the summation is taken over neighboring pairs (i, j).

In thermal equilibrium the distribution of the spins, say π, follows the Boltzmann law: $\pi(\mathbf{s}) \propto \exp(-E(\mathbf{s})/T)$, where T is a fixed temperature. In other words, we have

$$\pi(\mathbf{s}) = \frac{\mathrm{e}^{\frac{1}{T}\sum_{i \leftrightarrow j} s_i s_j}}{\mathcal{Z}},$$

where $\mathcal{Z}$ is the normalization constant, called the *partition function*. Apart from $\mathcal{Z}$, particular quantities of interest are the *mean energy per spin* $\mathbb{E}_\pi[\sum_{i \leftrightarrow j} S_i S_j / n^2]$ and the *mean magnetization per spin* $\mathbb{E}_\pi[\sum_i S_i / n^2]$. These quantities can be obtained via Monte Carlo simulation, provided that one can sample efficiently from the target distribution π (see below).

In Figure 6.7 a sample from π is given (black = 1, white = -1) for $n = 30$ at the so-called *critical temperature* $T = 2/\ln(1+\sqrt{2}) \approx 2.269$.

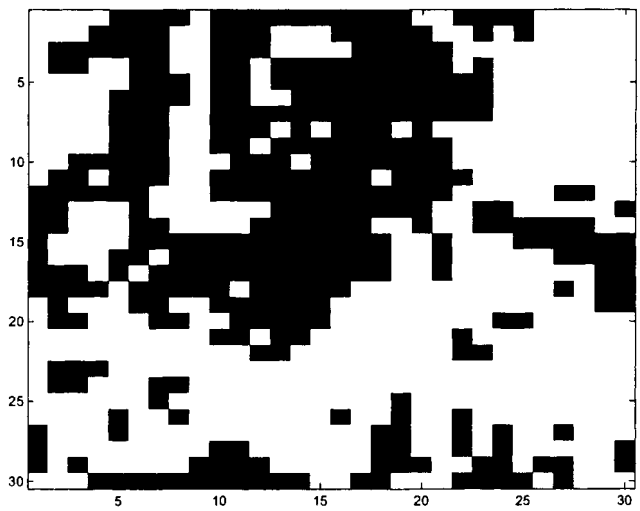


Figure 6.7 Ising configuration at the critical temperature.

We next define the *Potts model* — which can be viewed as a generalization of the Ising model — and explain how to generate samples from this extended model and thus, in particular, how to generate Figure 6.7.

6.5.2 Potts Model

Let $\{1, \ldots, J\}$ be an enumeration of spatial positions (sites), and let ψ_{ij} be some symmetrical and positive function relating the sites to each other, for example,

$$\psi_{ij} = \begin{cases} \beta\,(>0) & \text{if } i \text{ and } j \text{ are neighbors} \\ 0 & \text{otherwise.} \end{cases} \tag{6.13}$$

Assign to each site i a "color" x_i. Suppose there are K such colors, labeled $\{1, \ldots, K\}$. Define $\mathbf{x} = (x_1, \ldots, x_J)$ and let $\mathscr{X}$ be the space of such configurations. On $\mathscr{X}$ we define

the target pdf $f(\mathbf{x}) \propto \mathrm{e}^{H(\mathbf{x})}$ with

$$H(\mathbf{x}) = \sum_{i<j} \psi_{ij}\, I_{\{x_i = x_j\}} \,.$$

To see that the Ising model is a special case of the Potts model, define $x_i = I_{\{s_i=1\}}$ and ψ_{ij} as in (6.13), with $\beta = 4/T$. Then

$$\frac{1}{T}\sum_{i\leftrightarrow j} s_i\, s_j = \frac{1}{T}\sum_{i\leftrightarrow j} 2\left(I_{\{x_i=x_j\}} - \frac{1}{2}\right) = \sum_{i<j} \psi_{ij}\, I_{\{x_i=x_j\}} + \text{const}\,,$$

so that $\pi(\mathbf{s}) = f(\mathbf{x})$.

Next, we show how to generate a sample from the target pdf $f(\mathbf{x})$. To do so, we define auxiliary random variables $Y_{ij}, 1 \leqslant i < j \leqslant J$, such that conditional on $\mathbf{X} = \mathbf{x}$ the $\{Y_{ij}\}$ are independent, and each Y_{ij} is uniformly distributed on the interval $[0, a_{ij}]$, with $a_{ij} = \exp(\psi_{ij}\, I_{\{x_i=x_j\}}) \geqslant 1$. In other words, the conditional pdf of $\mathbf{Y} = \{Y_{ij}\}$ given $\mathbf{X} = \mathbf{x}$ is

$$f(\mathbf{y} \,|\, \mathbf{x}) = \prod_{i<j} \frac{I_{\{y_{ij} \leqslant a_{ij}\}}}{a_{ij}} = \prod_{i<j} I_{\{y_{ij} \leqslant a_{ij}\}}\, \mathrm{e}^{-H(\mathbf{x})} \,.$$

The significance of this is that the joint pdf of $\mathbf{X}$ and $\mathbf{Y}$ is now simply

$$f(\mathbf{x},\mathbf{y}) = f(\mathbf{x})\, f(\mathbf{y}|\mathbf{x}) \propto \begin{cases} 1 & \text{if } y_{ij} \leqslant a_{ij}, \text{ for all } i<j\,, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, $(\mathbf{X}, \mathbf{Y})$ is *uniformly* distributed. More importantly, because $f(\mathbf{x} \,|\, \mathbf{y}) \propto f(\mathbf{x}, \mathbf{y})$, we find that $\mathbf{X} \,|\, \mathbf{y}$ is uniformly distributed over the set $\mathscr{A} = \{\mathbf{x} : y_{ij} \leqslant \exp(\psi_{ij}\, I_{\{x_i=x_j\}})$ for all $i < j\}$. Now, either $y_{ij} \in [0,1]$ or $y_{ij} \in (1, \mathrm{e}^{\psi_{ij}}]$. In the former case, for any $\mathbf{x} \in \mathscr{A}$, the coordinates x_i and x_j range over all the colors, and, by the uniformity, each color is equally likely. But in the latter case, x_i must be equal to x_j. Thus, for a given $\mathbf{y}$, the sites i, j (with $i < j$) for which $y_{ij} > 1$ can be gathered into clusters, and within each such cluster the sites have identical colors. Moreover, given $\mathbf{y}$, the colors within the clusters are independent and uniformly distributed on $\{1, \ldots, K\}$. The same holds for the colors of the remaining positions, which can be viewed as one-cluster sites.

Hence, we can easily generate both $\mathbf{X} \,|\, \mathbf{y}$ and $\mathbf{Y} \,|\, \mathbf{x}$. As a consequence, we can use the Gibbs sampler to (approximately) sample from $f(\mathbf{x}, \mathbf{y})$; that is, we iteratively sample from $f(\mathbf{x} \,|\, \mathbf{y})$ and $f(\mathbf{y} \,|\, \mathbf{x})$. Finally, to obtain a sample $\mathbf{X}$ from $f(\mathbf{x})$, we generate $(\mathbf{X}, \mathbf{Y})$ via the Gibbs sampler and simply ignore $\mathbf{Y}$.

To simplify matters further, note that instead of the exact value Y_{ij} it suffices to know only the variable $B_{ij} = I_{\{Y_{ij} \geqslant 1\}}$. Given $\mathbf{X} = \mathbf{x}$, B_{ij} has a $\mathrm{Ber}(1 - \mathrm{e}^{-\psi_{ij}})$ distribution if $x_i = x_j$, and $B_{ij} = 0$ otherwise. This leads to the following so-called Swendsen–Wang algorithm.

Algorithm 6.5.1 (Swendsen–Wang)

1. *Given* $\{X_i\}$, *generate* $B_{ij} \sim \mathrm{Ber}(I_{\{X_i=X_j\}}(1 - e^{-\psi_{ij}}))$ *for* $1 \leqslant i < j \leqslant J$.

2. *Given* $\{B_{ij}\}$, *generate* $X_i, i = 1, \ldots, J$ *by clustering all the sites and choosing each cluster color independently and uniformly from* $\{1, \ldots, K\}$.

Remark 6.5.1 (Data Augmentation) The above idea of introducing an *auxiliary* variable $\mathbf{y}$ to make sampling from $f(\mathbf{x})$ easier is also known as *data augmentation*. The composition method described in Section 2.3.3 can be viewed as another example of data augmentation. Namely, suppose we want to sample from the mixture pdf

$$f(x) = \sum_{i=1}^{K} p_i f_i(x) .$$

Let Y be the discrete random variable taking values in $\{1, \ldots, K\}$ corresponding to the probabilities $\{p_i\}$. Using the composition method, it is easy to sample from the joint pdf of X and Y: first, draw Y according to $\{p_i\}$ and then sample X conditional on $Y = i$; that is, sample from $f_i(x)$. By simply ignoring Y, we obtain a sample from $f(x)$.

6.6 BAYESIAN STATISTICS

One of the main application areas of the MCMC method is Bayesian statistics. The mainstay of the Bayesian approach is Bayes' rule (1.6), which, in terms of pdfs, can be written as

$$f(\mathbf{y} \mid \mathbf{x}) = \frac{f(\mathbf{x} \mid \mathbf{y})\, f(\mathbf{y})}{\int f(\mathbf{x} \mid \mathbf{y})\, f(\mathbf{y})\, d\mathbf{y}} \propto f(\mathbf{x} \mid \mathbf{y})\, f(\mathbf{y}) . \tag{6.14}$$

In other words, for any two random variables $\mathbf{X}$ and $\mathbf{Y}$, the conditional distribution of $\mathbf{Y}$ given $\mathbf{X} = \mathbf{x}$ is proportional to the product of the conditional pdf of $\mathbf{X}$ given $\mathbf{Y} = \mathbf{y}$ and the pdf of $\mathbf{Y}$. Note that instead of writing f_X, f_Y, $f_{X \mid Y}$, and $f_{Y \mid X}$ in the formula above, we have used the *same letter* f for the pdf of $\mathbf{X}$, $\mathbf{Y}$, and the conditional pdfs. This particular style of notation is typical in Bayesian analysis and can be of great descriptive value, despite its apparent ambiguity. We will use this notation whenever we work in a Bayesian setting.

The significance of (6.14) becomes clear when it is employed in the context of Bayesian parameter estimation, sometimes referred to as *Bayesian learning*. The following example explains the ideas.

■ **EXAMPLE 6.7 Coin Flipping and Bayesian Learning**

Consider the basic random experiment in Example 1.1 on page 3, where we toss a biased coin n times. Suppose that the outcomes are $x_1, \ldots, x_n$, with $x_i = 1$ if the i-th toss is heads and $x_i = 0$ otherwise, $i = 1, \ldots, n$. Let p denote the probability of heads. We want to obtain information about p from the data $\mathbf{x} = (x_1, \ldots, x_n)$, for example, construct a CI.

The crucial idea is to summarize the information about p via a probability density $f(p)$. For example, if we know nothing about p, we take $f(p)$ uniformly distributed on the $(0, 1)$ interval, that is, $f(p) = 1, 0 \leqslant p \leqslant 1$. In effect, we treat p as a random variable. Now, obviously, the data $\mathbf{x}$ will affect our knowledge of p, and the way to update this information is to use Bayes' formula:

$$f(p \mid \mathbf{x}) \propto f(\mathbf{x} \mid p)\, f(p) .$$

The density $f(p)$ is called the *prior* density; $f(p \mid \mathbf{x})$ is called the *posterior density*; and $f(\mathbf{x} \mid p)$ is referred to as the *likelihood*. In our case, given p, the $\{X_i\}$ are independent and $\mathrm{Ber}(p)$ distributed, so that

$$f(\mathbf{x} \mid p) = \prod_{i=1}^{n} p^{x_i} (1-p)^{1-x_i} = p^s (1-p)^{n-s} ,$$

with $s = x_1 + \cdots + x_n$ representing the total number of successes. Thus, using a uniform prior ($f(p) = 1$) gives a posterior pdf

$$f(p \mid \mathbf{x}) = c\, p^s (1-p)^{n-s},$$

which is the pdf of the Beta$(s+1, n-s+1)$ distribution. The normalization constant is $c = (n+1)\binom{n}{s}$.

A Bayesian CI for p is now formed by taking the appropriate quantiles of the posterior pdf. As an example, suppose that $n = 100$ and $s = 1$. Then, a left one-sided 95% CI for p is $[0, 0.0461]$, where 0.0461 is the 0.95 quantile of the Beta$(2, 100)$ distribution. As an estimate for p, one often takes the value for which the pdf is maximal, the so-called *mode* of the pdf. In this case, the mode is 0.01, coinciding with the sample mean. Figure 6.8 gives a plot of the posterior pdf for this problem.

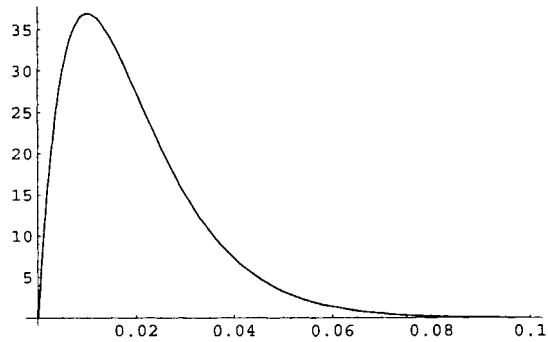


Figure 6.8 Posterior pdf for p, with $n = 100$ and $s = 1$.

Generalizing the previous example, a typical situation where MCMC (in particular Gibbs sampling) can be used in Bayesian statistics is the following. Suppose we want to sample from a posterior density $f(\boldsymbol{\theta} \mid \mathbf{x})$, where the data $\mathbf{x}$ are given (fixed) and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$ is the parameter of interest. Suppose that it is easy to sample from $f(\theta_i \mid \theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_k, \mathbf{x})$ for all i. Then we can use the Gibbs sampler to obtain a sample Θ from $f(\boldsymbol{\theta} \mid \mathbf{x})$. The following example, adapted from Gelman et al. [5], illustrates the general idea.

■ EXAMPLE 6.8 Poisson Disruption Problem

Suppose the random variables $X_1, \ldots, X_n$ describe the number of disasters in n subsequent years. In some random year K the rate of disasters changes from λ_1 to λ_2. Such a K is often called a *change point*. Our prior knowledge of λ_i is summarized by a Gamma(a_i, η_i), where shape parameter a_i is known. In turn, η_i is given by a Gamma(b_i, c_i) distribution, where both b_i and c_i are known. Let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2)$ and $\boldsymbol{\eta} = (\eta_1, \eta_2)$. We are given the data $\mathbf{x} = (x_1, \ldots, x_n)$, and the objective is to simulate from the posterior distribution of $\boldsymbol{\theta} = (\lambda_1, \lambda_2, \eta_1, \eta_2, K)$ given $\mathbf{x}$.

For the model we have the following hierarchical structure:

1. K has some discrete pdf $f(K)$ on $1, \ldots, n$.
2. Given K, the $\{\eta_i\}$ are independent and have a Gamma(b_i, c_i) distribution for $i = 1, 2$.

3. Given K and $\boldsymbol{\eta}$, the $\{\lambda_i\}$ are independent and have a $\mathsf{Gamma}(a_i, \eta_i)$ distribution for $i = 1, 2$.

4. Given $K, \boldsymbol{\eta}$, and $\boldsymbol{\lambda}$, the $\{X_i\}$ are independent and have a $\mathsf{Poi}(\lambda_1)$ distribution for $i = 1, \ldots, K$, and a $\mathsf{Poi}(\lambda_2)$ distribution for $i = K+1, \ldots, n$.

It follows from point 4. that

$$
\begin{aligned}
f(\mathbf{x} \mid \boldsymbol{\lambda}, \boldsymbol{\eta}, K) &= \prod_{i=1}^{K} \mathrm{e}^{-\lambda_1} \frac{\lambda_1^{x_i}}{x_i!} \prod_{i=K+1}^{n} \mathrm{e}^{-\lambda_2} \frac{\lambda_2^{x_i}}{x_i!} \\
&= \mathrm{e}^{-\lambda_1 K} \lambda_1^{\sum_{i=1}^{K} x_i} \mathrm{e}^{-\lambda_2 (n-K)} \lambda_2^{\sum_{i=K+1}^{n} x_i} \prod_{i=1}^{n} \frac{1}{x_i!} \,.
\end{aligned}
$$

Moreover, by the product rule (1.4), the joint pdf is given by

$$
\begin{aligned}
f(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\eta}, K) \propto\; & f(K) \, \mathrm{e}^{-\lambda_1 K} \lambda_1^{\sum_{i=1}^{K} x_i} \mathrm{e}^{-\lambda_2 (n-K)} \lambda_2^{\sum_{i=K+1}^{n} x_i} \prod_{i=1}^{n} \frac{1}{x_i!} \\
& \times \mathrm{e}^{-\eta_1 \lambda_1} \lambda_1^{a_1 - 1} \eta_1^{a_1} \times \mathrm{e}^{-\eta_2 \lambda_2} \lambda_2^{a_2 - 1} \eta_2^{a_2} \\
& \times \mathrm{e}^{-c_1 \eta_1} \eta_1^{b_1 - 1} c_1^{b_1} \times \mathrm{e}^{-c_2 \eta_2} \eta_2^{b_2 - 1} c_2^{b_2} \,.
\end{aligned}
$$

As a consequence,

$$
f(\lambda_1 \mid \lambda_2, \boldsymbol{\eta}, K, \mathbf{x}) \propto \mathrm{e}^{-\lambda_1 (K + \eta_1)} \lambda_1^{a_1 - 1 + \sum_{i=1}^{K} x_i} \,.
$$

In other words, $(\lambda_1 \mid \lambda_2, \boldsymbol{\eta}, K, \mathbf{x}) \sim \mathsf{Gamma}(a_1 + \sum_{i=1}^{K} x_i,\ K + \eta_1)$. In a similar way, we have

$$
\begin{aligned}
(\lambda_2 \mid \lambda_1, \boldsymbol{\eta}, K, \mathbf{x}) &\sim \mathsf{Gamma}\left(a_2 + \textstyle\sum_{i=K+1}^{n} x_i,\ n - K + \eta_2\right) \,, \\
(\eta_1 \mid \boldsymbol{\lambda}, \eta_2, K, \mathbf{x}) &\sim \mathsf{Gamma}(a_1 + b_1,\ \lambda_1 + c_1) \,, \\
(\eta_2 \mid \boldsymbol{\lambda}, \eta_1, K, \mathbf{x}) &\sim \mathsf{Gamma}(a_2 + b_2,\ \lambda_2 + c_2) \,, \\
f(K \mid \boldsymbol{\lambda}, \boldsymbol{\eta}, \mathbf{x}) &\propto f(K) \, \mathrm{e}^{-K(\lambda_1 + \lambda_2)} \, (\lambda_1 / \lambda_2)^{\sum_{i=1}^{K} x_i} \,,
\end{aligned}
$$

so that Gibbs sampling may be used to sample from the posterior pdf $f(\boldsymbol{\lambda}, \boldsymbol{\eta}, K \mid \mathbf{x})$.

6.7 * OTHER MARKOV SAMPLERS

There exist many variants of the Metropolis–Hastings and Gibbs samplers. However, all of the known MCMC algorithms can be described via the following framework. Consider a Markov chain $\{(\mathbf{X}_n, \mathbf{Y}_n), n = 0, 1, 2, \ldots\}$ on the set $\mathscr{X} \times \mathscr{Y}$, where $\mathscr{X}$ is the target set and $\mathscr{Y}$ is an auxiliary set. Let $f(\mathbf{x})$ be the target pdf. Each transition of the Markov chain consists of two parts. The first is $(\mathbf{x}, \tilde{\mathbf{y}}) \to (\mathbf{x}, \mathbf{y})$, according to a transition matrix $\mathbf{Q}$; the second is $(\mathbf{x}, \mathbf{y}) \to (\mathbf{x}', \mathbf{y}')$, according to a transition matrix $\mathbf{R}$. In other words, the transition matrix $\mathbf{P}$ of the Markov chain is given by the product $\mathbf{Q}\,\mathbf{R}$. Both steps are illustrated in Figure 6.9 and further explained below.

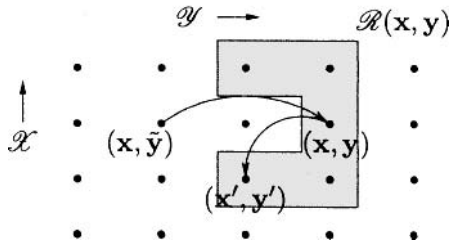


Figure 6.9 Each transition of the Markov chain consists of two steps: the Q-step, followed by the R-step.

The first step, the *Q-step*, changes the y-coordinate but leaves the x-coordinate intact. In particular, $\mathbf{Q}$ is of the form $\mathbf{Q}[(\mathbf{x},\tilde{\mathbf{y}}),(\mathbf{x},\mathbf{y})] = \mathbf{Q}_{\mathbf{x}}(\tilde{\mathbf{y}},\mathbf{y})$, where $\mathbf{Q}_{\mathbf{x}}$ is a transition matrix on $\mathscr{Y}$. Let $q_{\mathbf{x}}$ be a stationary distribution for $\mathbf{Q}_{\mathbf{x}}$, assuming that it exists.

The second step, the *R-step*, is determined by (a) the stationary distribution $q_{\mathbf{x}}$ and (b) a neighborhood structure on the set $\mathscr{X} \times \mathscr{Y}$. Specifically, we define for each point $(\mathbf{x},\mathbf{y})$ a set of neighbors $\mathscr{R}(\mathbf{x},\mathbf{y})$ such that *if* $(\mathbf{x}',\mathbf{y}')$ *is a neighbor of* $(\mathbf{x},\mathbf{y})$ *then the converse is also true*; see Figure 6.9, where the shaded area indicates the neighborhood set of $(\mathbf{x},\mathbf{y})$. The crucial step is now to define the transition matrix $\mathbf{R}$ as

$$\mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')] = c(\mathbf{x},\mathbf{y})\, f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}') \quad \text{for all} \quad (\mathbf{x}',\mathbf{y}') \in \mathscr{R}(\mathbf{x},\mathbf{y}) \;,$$

where $c(\mathbf{x},\mathbf{y}) = \sum_{(\mathbf{x}',\mathbf{y}')\in\mathscr{R}(\mathbf{x},\mathbf{y})} f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}')$. Note that $c(\mathbf{x},\mathbf{y}) = c(\mathbf{x}',\mathbf{y}')$ when $(\mathbf{x},\mathbf{y})$ and $(\mathbf{x}',\mathbf{y}')$ belong to the same neighborhood set. With this choice of $\mathbf{Q}$ and $\mathbf{R}$ it can be shown (see Problem 6.15) that the Markov chain has a stationary distribution

$$\mu(\mathbf{x},\mathbf{y}) = f(\mathbf{x})\, q_{\mathbf{x}}(\mathbf{y}) \;, \tag{6.15}$$

which is also the limiting distribution, provided that the chain is irreducible and aperiodic. In particular, by ignoring the y-coordinate, we see that the limiting pdf of $\mathbf{X}_n$ is the required target $f(\mathbf{x})$. This leads to the following *generalized Markov sampler* [11].

Algorithm 6.7.1 (Generalized Markov Sampler) Starting with an arbitrary $(\mathbf{X}_0,\mathbf{Y}_0)$, perform the following steps iteratively:

[***Q*-step:**] Given $(\mathbf{X}_n,\mathbf{Y}_n)$, generate $\mathbf{Y}$ from $\mathbf{Q}_{\mathbf{x}}(\mathbf{Y}_n,\mathbf{y})$.

[***R*-step:**] Given $\mathbf{Y}$ generate $(\mathbf{X}_{n+1},\mathbf{Y}_{n+1})$ from $\mathbf{R}[(\mathbf{X}_n,\mathbf{Y}),(\mathbf{x},\mathbf{y})]$.

Remark 6.7.1 Denoting $\mathscr{R}^-(\mathbf{x},\mathbf{y}) = \mathscr{R}(\mathbf{x},\mathbf{y}) \setminus \{(\mathbf{x},\mathbf{y})\}$, the sampler can be generalized further (see [11]) by redefining $\mathbf{R}$ as

$$\mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')] = \begin{cases} s(\mathbf{x},\mathbf{y})\, c(\mathbf{x},\mathbf{y})\, f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}') & \text{if } (\mathbf{x}',\mathbf{y}') \in \mathscr{R}^-(\mathbf{x},\mathbf{y}) \\ 1 - \displaystyle\sum_{(\mathbf{z},\mathbf{k})\in\mathscr{R}^-(\mathbf{x},\mathbf{y})} \mathbf{R}[(\mathbf{x},\mathbf{y}),(\mathbf{z},\mathbf{k})] & \text{if } (\mathbf{x}',\mathbf{y}') = (\mathbf{x},\mathbf{y}) \;, \end{cases} \tag{6.16}$$

where s is an arbitrary function such that, first, $s(\mathbf{x},\mathbf{y}) = s(\mathbf{x}',\mathbf{y}')$ for all $(\mathbf{x}',\mathbf{y}') \in \mathscr{R}(\mathbf{x},\mathbf{y})$ and, second, the quantities above are indeed probabilities.

The generalized Markov sampler framework makes it possible to obtain many different samplers in a simple and unified manner. We give two examples: the slice sampler and the reversible jump sampler.

6.7.1 Slice Sampler

Suppose we wish to generate samples from the pdf

$$f(\mathbf{x}) = b \prod_{k=1}^{m} f_k(\mathbf{x}) \,, \tag{6.17}$$

where b is a known or unknown constant and the $\{f_k\}$ are known positive functions — not necessarily densities. We employ Algorithm 6.7.1, where at the Q-step we generate, for a given $\mathbf{X} = \mathbf{x}$, a vector $\mathbf{Y} = (Y_1, \ldots, Y_m)$ by independently drawing each component Y_k from the uniform distribution on $[0, f_k(\mathbf{x})]$. Thus, $q_{\mathbf{x}}(\mathbf{y}) = 1/\prod_{k=1}^{m} f_k(\mathbf{x}) = b/f(\mathbf{x})$. Second, we let $\mathscr{R}(\mathbf{x}, \mathbf{y}) = \{(\mathbf{x}', \mathbf{y}) : f_k(\mathbf{x}') \geqslant y_k,\ k = 1, \ldots, m\}$. Then, (note that $f(\mathbf{x}')\, q_{\mathbf{x}'}(\mathbf{y}) = b$)

$$\mathbf{R}[(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y})] = \frac{1}{|\mathscr{R}(\mathbf{x}, \mathbf{y})|} \,.$$

In other words, in the R-step, given $\mathbf{x}$ and $\mathbf{y}$, we draw $\mathbf{X}'$ uniformly from the set $\{\mathbf{x}' : f_k(\mathbf{x}') \geqslant y_k, k = 1, \ldots, m\}$. This gives the following *slice sampler.*

Algorithm 6.7.2 (Slice Sampler)

Let $f(\mathbf{x})$ be of the form (6.17).

1. *Initialize $\mathbf{X}_1$. Set $t = 1$.*
2. *For $k = 1, \ldots, m$ draw $U_k \sim \mathsf{U}(0, 1)$ and let $Y_k = U_k\, f_k(\mathbf{X}_t)$.*
3. *Draw $\mathbf{X}_{t+1}$ uniformly from the set $\{\mathbf{x} : f_k(\mathbf{x}) \geqslant Y_k, k = 1, \ldots, m\}$.*
4. *Stop if a stopping criterion is met; otherwise, set $t = t + 1$ and repeat from Step 2.*

■ **EXAMPLE 6.9 Slice Sampler**

Suppose we want to generate a sample from the target pdf

$$f(x) = c\, \frac{x\, \mathrm{e}^{-x}}{1 + x}, \quad x \geqslant 0$$

using the slice sampler with $f_1(x) = x/(1 + x)$ and $f_2(x) = \mathrm{e}^{-x}$.

Suppose that at iteration t, $X_{t-1} = z$, and u_1 and u_2 are generated in Step 2. In Step 3, X_t is drawn uniformly from the set $\{x : f_1(x)/f_1(z) \geqslant u_1,\ f_2(x)/f_2(z) \geqslant u_2\}$, which implies the bounds $x \geqslant \frac{u_1 z}{1+z-u_1 z}$ and $x \leqslant z - \ln u_2$. Since for $z > 0$ and $0 \leqslant u_1, u_2 \leqslant 1$, the latter bound is larger than the former, the interval to be drawn from in Step 3 is $(\frac{u_1 z}{1+z-u_1 z},\ z - \ln u_2)$. Figure 6.10 depicts a histogram of $N = 10^5$ samples generated via the slice sampler, along with the true pdf $f(x)$. We see that the two are in close agreement.

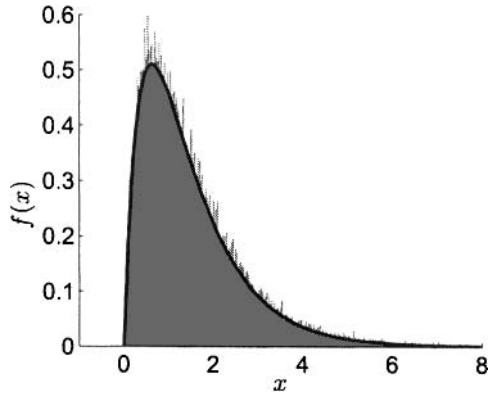


Figure 6.10 True density and histogram of samples produced by the slice sampler.

6.7.2 Reversible Jump Sampler

Reversible jump samplers [8] are useful for sampling from target spaces that contain vectors of different dimensions. This often occurs in Bayesian inference when different models for the data are considered.

■ **EXAMPLE 6.10 Regression Data**

Suppose some data $y_1, \ldots, y_n$ are the outcomes of independent random variables $\{Y_i\}$ of the form

$$Y_i = \sum_{j=0}^{M} \beta_j u_i^j + \varepsilon_i, \qquad \varepsilon_i \sim \mathsf{N}(0,1), \quad i = 1, \ldots, n\,, \tag{6.18}$$

where $u_1, \ldots, u_n$ are known variables, and $M \in \{0, \ldots, M_{\max}\}$ and the parameters $\{\beta_m\}$ are unknown. Let $\mathbf{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_M)$. Taking uniform (i.e., constant) priors for $\{\beta_m\}$ and M, we have the joint pdf

$$f(\mathbf{y}, m, \boldsymbol{\beta}) \propto \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{m}\beta_j u_i^j\right)^2\right]\,. \tag{6.19}$$

Denoting $\mathbf{x} = (m, \boldsymbol{\beta})$, the objective is to draw from the posterior pdf $f(\mathbf{x} \mid \mathbf{y}) = f(m, \boldsymbol{\beta} \mid \mathbf{y})$. This yields information not only about the parameters, but also about which model (expressed by M) is more appropriate. However, note that the dimensionality of $\mathbf{x}$ depends crucially on m, so that standard Gibbs or Metropolis–Hastings sampling is not appropriate.

The reversible jump sampler jumps between spaces of different dimensionality according to a set of allowed jumps (also called *moves*). In the above example one could, for instance, allow only jumps between vectors that differ in dimension by at most 1; that is, $\beta_0 \to \beta_0'$, $\beta_0 \to (\beta_0', \beta_1')$, $(\beta_0, \beta_1) \to \beta_0'$, and so on.

To formulate the reversible jump sampler in the generalized Markov sampler framework, define $\mathscr{Y} = \mathscr{X} \times \mathscr{M}$, where $\mathscr{M}$ is the set of moves; write a generic element as $(\mathbf{z}, m)$. In

the Q-step we take $\mathbf{Q}_{\mathbf{x}}(\cdot, (\mathbf{z}, m)) = p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})$. That is, a move of type m is selected according to some discrete pdf $p_{\mathbf{x}}(m)$. For example, the dimension of $\mathbf{x}$ is decreased, increased, or left unchanged. Then a new $\mathbf{z}$ is selected according to some transition function $q_m(\mathbf{x}, \mathbf{z})$. Note that the stationary pdf for the Q-step at $(\mathbf{z}, m)$ is $p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})$. The R-step is determined by defining $\mathscr{R}(\mathbf{x}, (\mathbf{z}, m)) = \{(\mathbf{x}, (\mathbf{z}, m)), (\mathbf{z}, (\mathbf{x}, m'))\}$, where m' is the reverse move of m, that is, from $\mathbf{z}$ to $\mathbf{x}$. Then (6.16) reduces to

$$\mathbf{R}[(\mathbf{x}, (\mathbf{z}, m)), (\mathbf{z}, (\mathbf{x}, m'))] = \frac{s(\mathbf{x}, (\mathbf{z}, m))}{1 + 1/\varrho}\,, \tag{6.20}$$

with $\varrho = \frac{f(\mathbf{z})\, p_{\mathbf{z}}(m')\, q_{m'}(\mathbf{z}, \mathbf{x})}{f(\mathbf{x})\, p_{\mathbf{x}}(m)\, q_m(\mathbf{x}, \mathbf{z})}$. Taking $s(\mathbf{x}, (\mathbf{z}, m)) = \min\{1 + \varrho,\ 1 + 1/\varrho\}$, the right-hand side of (6.20) reduces further to $\min\{\varrho, 1\}$. The transition $(\mathbf{x}, (\mathbf{z}, m)) \to (\mathbf{z}, (\mathbf{x}, m'))$ may thus be interpreted as acceptance of the proposed element $\mathbf{z}$. In effect, $\mathbf{Q}$ is used to propose a new element in accordance with the move m and transition function q, and $\mathbf{R}$ is used to accept or reject it in accordance with the above acceptance ratio. The reversible jump sampler may thus be viewed as a generalization of the Metropolis–Hastings sampler. This gives the following algorithm (to be iterated).

Algorithm 6.7.3 (Reversible Jump Sampler)
Given the current state $\mathbf{X}_t$*:*

1. *Generate* $m \sim p_{\mathbf{X}_t}(m)$.
2. *Generate* $\mathbf{Z} \sim q_m(\mathbf{X}_t, \mathbf{z})$. *Let* m' *be the reverse move, that is, from* $\mathbf{Z}$ *to* $\mathbf{X}_t$.
3. *Generate* $U \sim \mathrm{U}(0, 1)$ *and deliver*

$$\mathbf{X}_{t+1} = \begin{cases} \mathbf{Z}, & \text{if } U \le \alpha \\ \mathbf{X}_t, & \text{otherwise}\,, \end{cases} \tag{6.21}$$

where

$$\alpha = \min\left\{ \frac{f(\mathbf{Z})\, p_{\mathbf{Z}}(m')\, q_{m'}(\mathbf{Z}, \mathbf{X}_t)}{f(\mathbf{X}_t)\, p_{\mathbf{X}_t}(m)\, q_m(\mathbf{X}_t, \mathbf{Z})},\ 1 \right\}\,. \tag{6.22}$$

Remark 6.7.2 (Dimension Matching) When dealing with continuous random variables it is important to ensure that the transition densities are properly defined. Suppose that $\dim(\mathbf{x}) = d$ and $\dim(\mathbf{z}) = d' > d$. A possible way to generate a transition $\mathbf{x} \to \mathbf{z}$ is to first draw a $(d' - d)$-dimensional random vector $\mathbf{U}$ according to some density $g(\mathbf{u})$ and then let $\mathbf{z} = \phi(\mathbf{x}, \mathbf{U})$ for some bijection ϕ. This is known as *dimension matching* — the dimension of $(\mathbf{x}, \mathbf{u})$ must match that of $\mathbf{z}$. Note that by (1.20) the transition density is given by $q(\mathbf{x}, \mathbf{z}) = g(\mathbf{u})/|J_{(\mathbf{x},\mathbf{u})}(\phi)|$, where $|J_{(\mathbf{x},\mathbf{u})}(\phi)|$ is the absolute value of the determinant of the matrix of Jacobi of ϕ at $(\mathbf{x}, \mathbf{u})$.

■ **EXAMPLE 6.11 Example 6.10 (Continued)**

We illustrate the reversible jump sampler using regression data $\mathbf{y} = (y_1, \ldots, y_n)$ of the form (6.18), with $u_i = (i - 1)/20$, $i = 1, \ldots, 101$, $\beta_0 = 1$, $\beta_1 = 0.3$, and $\beta_2 = -0.2$. The data are depicted in Figure 6.11. Although it is obvious that a constant model ($m = 0$) does not fit the data, it is not clear if a linear model ($m = 1$) or a quadratic model ($m = 2$) is more appropriate. To assess the different models, we

can run a reversible jump sampler to produce samples from the posterior pdf $f(\mathbf{x} \mid \mathbf{y})$, which (up to a normalization constant) is given by the right-hand side of (6.19). A very basic implementation is the following:

Procedure

1. Initialize $\mathbf{X}_1 = \mathbf{x} = (m', \boldsymbol{\beta}')$. Set $t = 1$.

2. Choose $m \in \{0, 1, 2\}$ with equal probability.

3. Generate $\boldsymbol{\beta}$ from an $(m+1)$-dimensional normal pdf g_m with independent components, with means 0 and variances σ^2. Let $\mathbf{z} = (m, \boldsymbol{\beta})$.

4. Generate $U \sim \mathrm{U}(0,1)$. If

$$U \leqslant \min\left\{ \frac{f(\mathbf{z} \mid \mathbf{y})\ g_{m'}(\boldsymbol{\beta}')}{f(\mathbf{x} \mid \mathbf{y})\ g_m(\boldsymbol{\beta})},\ 1 \right\},$$

   set $\mathbf{X}_{t+1} = \mathbf{z}$; otherwise, set $\mathbf{X}_{t+1} = \mathbf{x}$.

5. If $t = N$ stop; otherwise, set $t = t + 1$, $\mathbf{x} = (m', \boldsymbol{\beta}') = \mathbf{X}_t$, and repeat from Step 2.

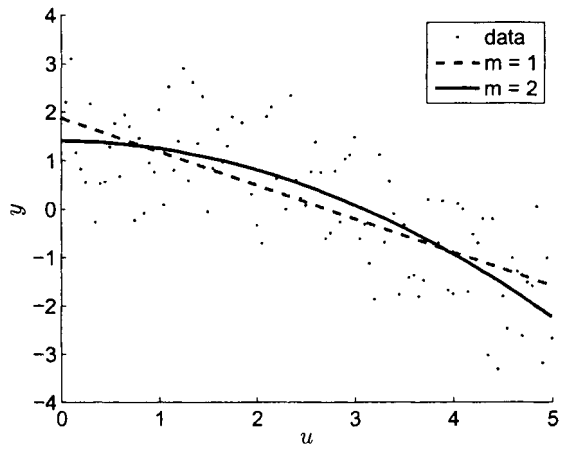


Figure 6.11 Regression data and fitted curves.

The above procedure, with $N = 10^5$ and $\sigma = 2$, produced 22,136 two-dimensional vectors $\boldsymbol{\beta}$ and 77,834 three-dimensional ones, giving posterior probabilities 0.221 and 0.778 for models 1 and 2, respectively. The posterior probability for the constant model was negligible (0.0003). This indicates that the quadratic model has the best fit. The regression parameters $\boldsymbol{\beta}$ are estimated via the sample means of the $\{\boldsymbol{\beta}_t\}$ for $m_t = 1$ or 2 and are found to be $(1.874, -0.691)$ and $(1.404, -0.011, -0.143)$. The corresponding regression curves are depicted in Figure 6.11.

6.8 SIMULATED ANNEALING

Simulated annealing is a popular optimization technique based on MCMC. This technique uses MCMC sampling to find a mode of a density $f(\mathbf{x})$ (a point $\mathbf{x}^*$ where $f(\mathbf{x})$ is maximal). It involves defining a family of densities of the form $f_T(\mathbf{x}) \propto [f(\mathbf{x})]^{1/T}$, where the parameter T is called the *temperature* of the distribution. MCMC sampling is used to draw a single element $\mathbf{X}^{(k)}$ from f_{T_k} for successively lower temperatures $T_1, T_2, \ldots$. Each element $\mathbf{X}^{(k)}$ is used as the initial element of the next chain. As the temperature is reduced, the distributions become sharply peaked at the global maxima of f. Thus, the $\{\mathbf{X}^{(k)}\}$ converge to a point. They can converge to a local maximum, but this possibility is reduced by careful selection of successive temperatures. The sequence of temperatures, or *annealing schedule*, is therefore critical to the success of the method. A common choice for the annealing schedule is a geometric progression, starting with a specified initial temperature and multiplying by a *cooling factor* in the interval $(0, 1)$ after each iteration.

Simulated annealing can also be applied to nonprobabilistic optimization problems. Given an objective function $S(\mathbf{x})$, one defines a Boltzmann distribution via the density $f(\mathbf{x}) \propto \mathrm{e}^{-S(\mathbf{x})}$ or $f(\mathbf{x}) \propto \mathrm{e}^{S(\mathbf{x})}$, depending on whether the objective is to minimize or maximize S. Global optima of S are then obtained by searching for the mode of the Boltzmann distribution. We illustrate the method via two worked examples, one based on the Metropolis–Hastings sampler and the other on the Gibbs sampler.

■ EXAMPLE 6.12 Traveling Salesman Problem

The traveling salesman problem (TSP) can be formulated as follows. Consider a weighted graph G with n nodes, labeled $1, 2, \ldots, n$. The nodes represent cities, and the edges represent the roads between the cities. Each edge from i to j has weight or cost c_{ij}, representing the length of the road. The problem is to find the shortest *tour* that visits all the cities exactly once except the starting city, which is also the terminating city. An example is given in Figure 6.12, where the bold lines form a possible tour.

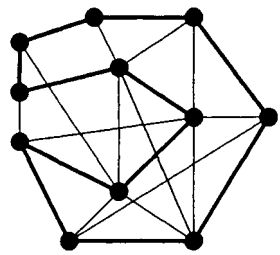


Figure 6.12 Find the shortest tour $\mathbf{x}$ visiting all nodes.

Without loss of generality, we may assume that the graph is *complete* (fully connected), because if it is not complete, we can always add some costs (distances) equal to $+\infty$. Let $\mathscr{X}$ be the set of all possible tours, and let $S(\mathbf{x})$ the total length of tour $\mathbf{x} \in \mathscr{X}$. We can represent each tour via a *permutation* of $(1, \ldots, n)$. For example, for $n = 4$, the permutation $(1, 3, 2, 4)$ represents the tour $1 \to 3 \to 2 \to 4 \to 1$. From now on, we identify a tour with its corresponding permutation. The objective

is thus to minimize

$$\min_{\mathbf{x}\in\mathscr{X}} S(\mathbf{x}) \quad = \quad \min_{\mathbf{x}\in\mathscr{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i,x_{i+1}} + c_{x_n,1} \right\} . \tag{6.23}$$

Note that the number of elements in $\mathscr{X}$ is typically very large, because $|\mathscr{X}| = n!$.

The TSP can be solved via simulated annealing in the following way. First, we define the target pdf to be the Boltzmann pdf $f(\mathbf{x}) = c\mathrm{e}^{-S(\mathbf{x})/T}$. Second, we define a neighborhood structure on the space of permutations $\mathscr{X}$ called *2-opt*. Here the neighbors of an arbitrary permutation $\mathbf{x}$ are found by (1) selecting two different indices from $\{1,\ldots,n\}$ and (2) reversing the path of $\mathbf{x}$ between those two indices. For example, if $\mathbf{x} = (1,2,\ldots,10)$ and indices 4 and 7 are selected, then $\mathbf{y} = (1,2,3,7,6,5,4,8,9,10)$; see Figure 6.13. Another example is: if $\mathbf{x} = (6,7,2,8,3,9,10,5,4,1)$ and indices 6 and 10 are selected, then $\mathbf{y} = (6,7,2,8,3,1,4,5,10,9)$.

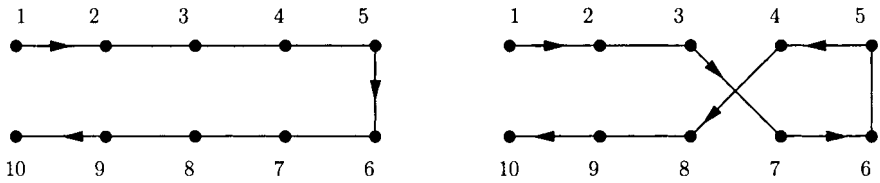


Figure 6.13 Illustration of the 2-opt neighborhood structure.

Third, we apply the Metropolis–Hastings algorithm to sample from the target. We need to supply a transition function $q(\mathbf{x},\mathbf{y})$ from $\mathbf{x}$ to one of its neighbors. Typically, the two indices for the 2-opt neighborhood are selected uniformly. This can be done, for example, by drawing a uniform permutation of $(1,\ldots,n)$ (see Section 2.8) and then selecting the first two elements of this permutation. The transition function is here constant: $q(\mathbf{x},\mathbf{y}) = q(\mathbf{y},\mathbf{x}) = 1/\binom{n}{2}$. It follows that in this case the acceptance probability is

$$\alpha = \min\left\{ \frac{f(\mathbf{y})}{f(\mathbf{x})}, 1 \right\} = \begin{cases} 1 & \text{if } S(\mathbf{y}) \leqslant S(\mathbf{x}) \\ \mathrm{e}^{-(S(\mathbf{y})-S(\mathbf{x}))/T} & \text{if } S(\mathbf{y}) > S(\mathbf{x}) \end{cases} . \tag{6.24}$$

By gradually decreasing the temperature T, the Boltzmann distribution becomes more and more concentrated around the global minimizer. This leads to the following generic simulated annealing algorithm with Metropolis–Hastings sampling.

Algorithm 6.8.1 (Simulated Annealing: Metropolis–Hastings Sampling)

1. *Initialize the starting state* $\mathbf{X}_0$ *and temperature* T_0. *Set* $t = 0$.
2. *Generate a new state* $\mathbf{Y}$ *from the symmetric proposal* $q(\mathbf{X}_t, \mathbf{y})$.
3. *If* $S(\mathbf{Y}) < S(\mathbf{X}_t)$ *let* $\mathbf{X}_{t+1} = \mathbf{Y}$. *If* $S(\mathbf{Y}) \geqslant S(\mathbf{X}_t)$, *generate* $U \sim \mathsf{U}(0,1)$ *and let* $\mathbf{X}_{t+1} = \mathbf{Y}$ *if*
$$U \leqslant e^{-(S(\mathbf{Y})-S(\mathbf{X}_t))/T_t};$$
otherwise, let $\mathbf{X}_{t+1} = \mathbf{X}_t$.
4. *Select a new temperature* $T_{t+1} \leqslant T_t$, *increase* t *by 1, and repeat from Step 2 until stopping.*

A common choice in Step 4 is to take $T_{t+1} = \beta T_t$ for some $\beta < 1$ close to 1, such as $\beta = 0.99$.

■ **EXAMPLE 6.13** n**-Queens Problem**

In the n-queens problem the objective is to arrange n queens on a $n \times n$ chess board in such a way that no queen can capture another queen. An illustration is given in Figure 6.14 for the case $n = 8$. Note that the configuration in Figure 6.14 does not solve the problem. We take $n = 8$ from now on. Note that each row of the chess board must contain exactly one queen. Denote the position of the queen in the i-th row by x_i; then each configuration can be represented by a vector $\mathbf{x} = (x_1, \ldots, x_8)$. For example, $\mathbf{x} = (2, 3, 7, 4, 8, 5, 1, 6)$ corresponds to the large configuration in Figure 6.14. Two other examples are given in the same figure. We can now formulate the problem of minimizing the function $S(\mathbf{x})$ representing the number of times the queens can capture each other. Thus $S(\mathbf{x})$ is the sum of the number of queens that can hit each other minus 1; see Figure 6.14, where $S(\mathbf{x}) = 2$ for the large configuration. Note that the minimal S value is 0. One of the optimal solutions is $\mathbf{x}^* = (5, 1, 8, 6, 3, 7, 2, 4)$.

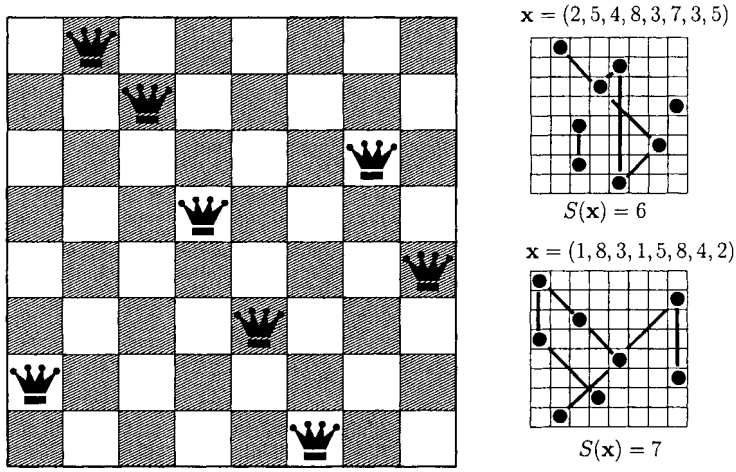


Figure 6.14 Position the eight queens such that no queen can capture another.

We show next how this optimization problem can be solved via simulated annealing using the Gibbs sampler. As in the previous TSP example, each iteration of the algorithm consists of sampling from the Boltzmann pdf $f(\mathbf{x}) = \mathrm{e}^{-S(\mathbf{x})/T}$ via the Gibbs sampler, followed by decreasing the temperature. This leads to the following generic simulated annealing algorithm using Gibbs sampling.

Algorithm 6.8.2 (Simulated Annealing: Gibbs Sampling)

1. *Initialize the starting state* $\mathbf{X}_0$ *and temperature* T_0. *Set* $t = 0$.
2. *For a given* $\mathbf{X}_t$, *generate* $\mathbf{Y} = (Y_1, \ldots, Y_n)$ *as follows:*
   - *i. Draw* Y_1 *from the conditional pdf* $f(x_1 \mid X_{t,2}, \ldots, X_{t,n})$.
   - *ii. Draw* Y_i *from* $f(x_i \mid Y_1, \ldots, Y_{i-1}, X_{t,i+1}, \ldots, X_{t,n})$, $i = 2, \ldots, n-1$.
   - *iii. Draw* Y_n *from* $f(x_n \mid Y_1, \ldots, Y_{n-1})$.
3. *Let* $\mathbf{X}_{t+1} = \mathbf{Y}$.
4. *If* $S(\mathbf{X}_t) = 0$ *stop and display the solution; otherwise, select a new temperature* $T_{t+1} \leqslant T_t$, *increase* t *by 1, and repeat from Step 2.*

Note that in Step 2 each Y_i is drawn from a discrete distribution on $\{1, \ldots, n\}$ with probabilities proportional to $\mathrm{e}^{-S(\mathbf{Z}_1)/T_t}, \ldots, \mathrm{e}^{-S(\mathbf{Z}_n)/T_t}$, where each $\mathbf{Z}_k$ is equal to the vector $(Y_1, \ldots, Y_{i-1}, k, X_{t,i+1}, \ldots, X_{t,n})$.

Other MCMC samplers can be used in simulated annealing. For example, in the *hide-and-seek* algorithm [20] the general hit-and-run sampler (Section 6.3) is used. Research motivated by the use of hit-and-run and discrete hit-and-run in simulated annealing, has resulted in the development of a theoretically derived cooling schedule that uses the recorded values obtained during the course of the algorithm to adaptively update the temperature [22, 23].

6.9 PERFECT SAMPLING

Returning to the beginning of this chapter, suppose that we wish to generate a random variable X taking values in $\{1, \ldots, m\}$ according to a target distribution $\pi = \{\pi_i\}$. As mentioned, one of the main drawbacks of the MCMC method is that each sample X_t is only *asymptotically* distributed according to π, that is, $\lim_{t \to \infty} \mathbb{P}(X_t = i) = \pi_i$. In contrast, *perfect sampling* is an MCMC technique that produces exact samples from π.

Let $\{X_t\}$ be a Markov chain with state space $\{1, \ldots, m\}$, transition matrix P, and stationary distribution π. We wish to generate the $\{X_t,\ t = 0, -1, -2, \ldots\}$ in such a way that X_0 has the desired distribution. We can draw X_0 from the m-point distribution corresponding to the X_{-1}-th row of P, see Algorithm 2.7.1. This can be done via the IT method, which requires the generation of a random variable $U_0 \sim \mathsf{U}(0,1)$. Similarly, X_{-1} can be generated from X_{-2} and $U_{-1} \sim \mathsf{U}(0,1)$. In general, we see that for any negative time $-t$ the random variable X_0 depends on X_{-t} and the independent random variables $U_{-t+1}, \ldots, U_0 \sim \mathsf{U}(0,1)$.

Next, consider m dependent copies of the Markov chain, starting from each of the states $1, \ldots, m$ and using the *same* random numbers $\{U_i\}$ — similar to the CRV method. Then, if two paths coincide, or *coalesce*, at some time, from that time on, both paths will be identical.

The paths are said to be *coupled.* The main point of the perfect sampling method is that if the chain is ergodic (in particular, if it is aperiodic and irreducible), then *with probability 1 there exists a negative time $-T$ such that all m paths will have coalesced before or at time 0.* The situation is illustrated in Figure 6.15.

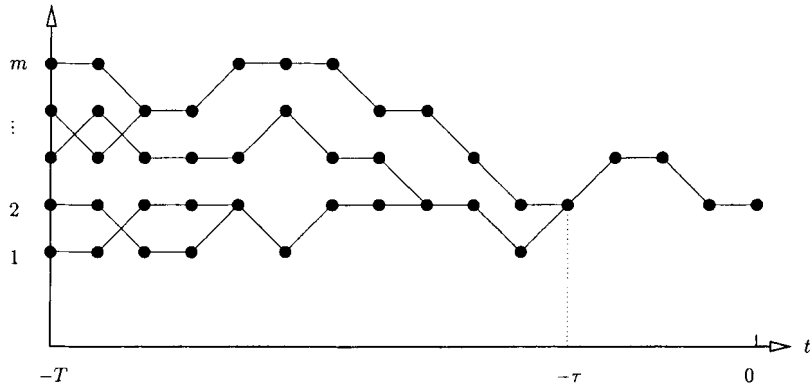


Figure 6.15 All Markov chains have coalesced at time $-\tau$.

Let $\mathbf{U}$ represent the vector of all $U_t, t \leqslant 0$. For each $\mathbf{U}$ we know there exists, with probability 1, a $-T(\mathbf{U}) < 0$ such that by time 0 all m coupled chains defined by $\mathbf{U}$ have coalesced. Moreover, if we start at time $-T$ a *stationary* version of the Markov chain, using again the same $\mathbf{U}$, this stationary chain must, at time $t = 0$, have coalesced with the other ones. Thus, any of the m chains has at time 0 the same distribution as the stationary chain, which is π.

Note that in order to construct T we do not need to know the whole (infinite vector) $\mathbf{U}$. Instead, we can work backward from $t = 0$ by generating U_{-1} first, and checking if $-T = -1$. If this is not the case, generate U_{-2} and check if $-T = -2$, and so on. This leads to the following algorithm, due to Propp and Wilson [18], called *coupling from the past.*

Algorithm 6.9.1 (Coupling from the Past)

1. *Generate* $U_0 \sim \mathsf{U}(0,1)$. *Set* $\mathbf{U}_0 = U_0$. *Set* $t = -1$.
2. *Generate* m *Markov chains, starting at* t *from each of the states* $1, \ldots, m$, *using the same random vector* $\mathbf{U}_{t+1}$.
3. *Check if all chains have coalesced before or at time 0. If so, return the common value of the chains at time 0 and stop; otherwise, generate* $U_t \sim \mathsf{U}(0,1)$, *let* $\mathbf{U}_t = (U_t, \mathbf{U}_{t+1})$, *set* $t = t - 1$, *and repeat from Step 2.*

Although perfect sampling seems indeed perfect in that it returns an exact sample from the target π rather than an approximate one, practical applications of the technique are, presently, quite limited. Not only is the technique difficult or impossible to use for most continuous simulation systems, it is also much more computationally intensive than simple MCMC.

PROBLEMS

6.1 Verify that the local balance equation (6.3) holds for the Metropolis–Hastings algorithm.

6.2 When running an MCMC algorithm, it is important to know when the transient (or *burn-in*) period has finished; otherwise, steady-state statistical analyses such as those in Section 4.3.2 may not be applicable. In practice this is often done via a visual inspection of the sample path. As an example, run the random walk sampler with normal target distribution $\mathsf{N}(10, 1)$ and proposal $Y \sim \mathsf{N}(x, 0.01)$. Take a sample size of $N = 5000$. Determine roughly when the process reaches stationarity.

6.3 A useful tool for examining the behavior of a stationary process $\{X_t\}$ obtained, for example, from an MCMC simulation, is the covariance function $R(t) = \mathrm{Cov}(X_t, X_0)$; see Example 6.4. Estimate the covariance function for the process in Problem 6.2 and plot the results. In Matlab's *signal processing* toolbox this is implemented under the M-function `xcov.m`. Try different proposal distributions of the form $\mathsf{N}(x, \sigma^2)$ and observe how the covariance function changes.

6.4 Implement the independence sampler with an $\mathsf{Exp}(1)$ target and an $\mathsf{Exp}(\lambda)$ proposal distribution for several values of λ. Similar to the importance sampling situation, things go awry when the sampling distribution gets too far from the target distribution, in this case when $\lambda > 2$. For each run, use a sample size of 10^5 and start with $x = 1$.

a) For each value $\lambda = 0.2, 1, 2$, and 5, plot a histogram of the data and compare it with the true pdf.

b) For each value of the above values of λ, calculate the sample mean and repeat this for 20 independent runs. Make a dotplot of the data (plot them on a line) and notice the differences. Observe that for $\lambda = 5$ most of the sample means are below 1, and thus underestimate the true expectation 1, but a few are significantly greater. Observe also the behavior of the corresponding auto-covariance functions, both between the different λs and, for $\lambda = 5$, within the 20 runs.

6.5 Implement the random walk sampler with an $\mathsf{Exp}(1)$ target distribution, where Z (in the proposal $Y = x + Z$) has a double exponential distribution with parameter λ. Carry out a study similar to that in Problem 6.4 for different values of λ, say $\lambda = 0.1, 1, 5, 20$. Observe that (in this case) the random walk sampler has a more stable behavior than the independence sampler.

6.6 Let $\mathbf{X} = (X, Y)^T$ be a random column vector with a bivariate normal distribution with expectation vector $\mathbf{0} = (0, 0)^T$ and covariance matrix

$$\Sigma = \begin{pmatrix} 1 & \varrho \\ \varrho & 1 \end{pmatrix}.$$

a) Show that $(Y \mid X = x) \sim \mathsf{N}(\varrho\, x, 1 - \varrho^2)$ and $(X \mid Y = y) \sim \mathsf{N}(\varrho\, y, 1 - \varrho^2)$.

b) Write a systematic Gibbs sampler to draw 10^4 samples from the bivariate distribution $\mathsf{N}(\mathbf{0}, \Sigma)$ and plot the data for $\varrho = 0, 0.7$ and 0.9.

6.7 A remarkable feature of the Gibbs sampler is that the conditional distributions in Algorithm 6.4.1 contain sufficient information to generate a sample from the joint one. The following result (by Hammersley and Clifford [9]) shows that it is possible to directly

express the joint pdf in terms of the conditional ones. Namely,

$$f(x,y) = \frac{f_{Y|X}(y|x)}{\int \frac{f_{Y|X}(y|x)}{f_{X|Y}(x|y)}\, dy} .$$

Prove this. Generalize this to the n-dimensional case.

6.8 In the Ising model the *expected magnetization per spin* is given by

$$M(T) = \frac{1}{n^2}\, \mathbb{E}_{\pi_T} \left[\sum_i S_i \right] ,$$

where π_T is the Boltzmann distribution at temperature T. Estimate $M(T)$, for example via the Swendsen–Wang algorithm, for various values of $T \in [0,5]$, and observe that the graph of $M(T)$ changes sharply around the critical temperature $T \approx 2.61$. Take $n = 20$ and use periodic boundaries.

6.9 Run Peter Young's Java applet in

`http://bartok.ucsc.edu/peter/java/ising/keep/ising.html`

to gain a better understanding of how the Ising model works.

6.10 As in Example 6.6, let $\mathscr{X}^* = \{\mathbf{x} : \sum_{i=1}^n x_i = m, x_i \in \{0,\ldots,m\}, i = 1,\ldots,n\}$. Show that this set has $\binom{m+n-1}{n-1}$ elements.

6.11 In a simple model for a closed queueing network with n queues and m customers, it is assumed that the service times are independent and exponentially distributed, say with rate μ_i for queue i, $i = 1,\ldots,n$. After completing service at queue i, the customer moves to queue j with probability p_{ij}. The $\{p_{ij}\}$ are the so-called *routing probabilities*.

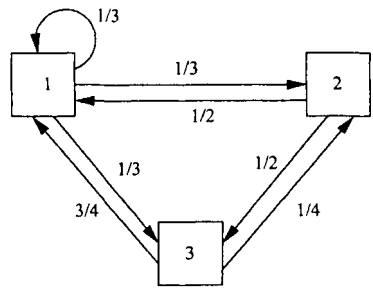


Figure 6.16 A closed queueing network.

It can be shown (see, for example, [12]) that the stationary distribution of the number of customers in the queues is of product form (6.10), with f_i being the pdf of the $\mathsf{G}(1 - y_i/\mu_i)$ distribution; thus, $f_i(x_i) \propto (y_i/\mu_i)^{x_i}$. Here the $\{y_i\}$ are constants that are obtained from the following set of *flow balance* equations:

$$y_i = \sum_j y_j\, p_{ji} , \quad i = 1,\ldots,n , \tag{6.25}$$

which has a one-dimensional solution space. Without loss of generality, y_1 can be set to 1 to obtain a unique solution.

Consider now the specific case of the network depicted in Figure 6.16, with $n = 3$ queues. Suppose the service rates are $\mu_1 = 2$, $\mu_2 = 1$, and $\mu_3 = 1$. The routing probabilities are given in the figure.

a) Show that a solution to (6.25) is $(y_1, y_2, y_3) = (1, 10/21, 4/7)$.

b) For $m = 50$ determine the exact normalization constant C.

c) Implement the procedure of Example 6.6 to estimate C via MCMC and compare the estimate for $m = 50$ with the exact value.

6.12 Let $X_1, \ldots, X_n$ be a random sample from the $\mathsf{N}(\mu, \sigma^2)$ distribution. Consider the following Bayesian model:

- $f(\mu, \sigma^2) = 1/\sigma^2$;
- $(\mathbf{x}_i \mid \mu, \sigma) \sim \mathsf{N}(\mu, \sigma^2)$, $i = 1, \ldots, n$ independently.

Note that the prior for (μ, σ^2) is *improper*. That is, it is not a pdf in itself, but by obstinately applying Bayes' formula, it does yield a proper posterior pdf. In some sense it conveys the least amount of information about μ and σ^2. Let $\mathbf{x} = (x_1, \ldots, x_n)$ represent the data. The posterior pdf is given by

$$f(\mu, \sigma^2 \mid \mathbf{x}) = \left(2\pi\sigma^2\right)^{-n/2} \exp\left\{-\frac{1}{2}\frac{\sum_i (x_i - \mu)^2}{\sigma^2}\right\} \frac{1}{\sigma^2}\,.$$

We wish to sample from this distribution via the Gibbs sampler.

a) Show that $(\mu \mid \sigma^2, \mathbf{x}) \sim \mathsf{N}(\bar{x},\ \sigma^2/n)$, where $\bar{x}$ is the sample mean.

b) Prove that

$$f(\sigma^2 \mid \mu, \mathbf{x}) \propto \frac{1}{(\sigma^2)^{n/2+1}} \exp\left(-\frac{n}{2}\frac{V_\mu}{\sigma^2}\right), \tag{6.26}$$

where $V_\mu = \sum_i (x_i - \mu)^2/n$ is the classical sample variance for known μ. In other words, $(1/\sigma^2 \mid \mu, \mathbf{x}) \sim \mathsf{Gamma}(n/2, nV_\mu/2)$.

c) Implement a Gibbs sampler to sample from the posterior distribution, taking $n = 100$. Run the sampler for 10^5 iterations. Plot the histograms of $f(\mu \mid \mathbf{x})$ and $f(\sigma^2 \mid \mathbf{x})$ and find the sample means of these posteriors. Compare them with the classical estimates.

d) Show that the true posterior pdf of μ given the data is given by

$$f(\mu \mid \mathbf{x}) \propto \left((\mu - \bar{x})^2 + V\right)^{-n/2},$$

where $V = \sum_i (x_i - \bar{x})^2/n$. (Hint: in order to evaluate the integral

$$f(\mu \mid \mathbf{x}) = \int_0^\infty f(\mu, \sigma^2 \mid \mathbf{x})\, d\sigma^2$$

write it first as $(2\pi)^{-n/2} \int_0^\infty t^{n/2-1} \exp(-\frac{1}{2}\, t\, c)\, dt$, where $c = n\, V_\mu$, by applying the change of variable $t = 1/\sigma^2$. Show that the latter integral is proportional to $c^{-n/2}$. Finally, apply the decomposition $V_\mu = (\bar{x} - \mu)^2 + V$.)

6.13 Suppose $f(\boldsymbol{\theta} \mid \mathbf{x})$ is the posterior pdf for some Bayesian estimation problem. For example, $\boldsymbol{\theta}$ could represent the parameters of a regression model based on the data $\mathbf{x}$. An important use for the posterior pdf is to make predictions about the distribution of other

random variables. For example, suppose the pdf of some random variable $\mathbf{Y}$ depends on $\boldsymbol{\theta}$ via the conditional pdf $f(\mathbf{y} \mid \boldsymbol{\theta})$. The *predictive pdf* of Y given $\mathbf{x}$ is defined as

$$f(\mathbf{y} \mid \mathbf{x}) = \int f(\mathbf{y} \mid \boldsymbol{\theta}) f(\boldsymbol{\theta} \mid \mathbf{x})\, d\boldsymbol{\theta} ,$$

which can be viewed as the expectation of $f(\mathbf{y} \mid \boldsymbol{\theta})$ under the posterior pdf. Therefore, we can use Monte Carlo simulation to approximate $f(\mathbf{y} \mid \mathbf{x})$ as

$$f(\mathbf{y} \mid \mathbf{x}) \approx \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{y} \mid \boldsymbol{\theta}_i) ,$$

where the sample $\{\boldsymbol{\theta}_i, i = 1, \ldots, N\}$ is obtained from $f(\boldsymbol{\theta} \mid \mathbf{x})$; for example, via MCMC.

As a concrete application, suppose that the independent measurement data: $-0.4326, -1.6656, 0.1253, 0.2877, -1.1465$ come from some $\mathsf{N}(\mu, \sigma^2)$ distribution. Define $\boldsymbol{\theta} = (\mu, \sigma^2)$. Let $Y \sim \mathsf{N}(\mu, \sigma^2)$ be a new measurement. Estimate and draw the predictive pdf $f(y \mid \mathbf{x})$ from a sample $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N$ obtained via the Gibbs sampler of Problem 6.12. Take $N = 10{,}000$. Compare this with the "common-sense" Gaussian pdf with expectation $\bar{x}$ (sample mean) and variance s^2 (sample variance).

6.14 In the *zero-inflated Poisson* (ZIP) model, random data $X_1, \ldots, X_n$ are assumed to be of the form $X_i = R_i Y_i$, where the $\{Y_i\}$ have a $\mathsf{Poi}(\lambda)$ distribution and the $\{R_i\}$ have a $\mathsf{Ber}(p)$ distribution, all independent of each other. Given an outcome $\mathbf{x} = (x_1, \ldots, x_n)$, the objective is to estimate both λ and p. Consider the following hierarchical Bayes model:

- $p \sim \mathsf{U}(0, 1)$ (prior for p),
- $(\lambda \mid p) \sim \mathsf{Gamma}(a, b)$ (prior for λ),
- $(r_i \mid p, \lambda) \sim \mathsf{Ber}(p)$ independently (from the model above),
- $(x_i \mid \mathbf{r}, \lambda, p) \sim \mathsf{Poi}(\lambda r_i)$ independently (from the model above),

where $\mathbf{r} = (r_1, \ldots, r_n)$ and a and b are known parameters. It follows that

$$f(\mathbf{x}, \mathbf{r}, \lambda, p) = \frac{b^a \lambda^{a-1} \mathrm{e}^{-b\lambda}}{\Gamma(a)} \prod_{i=1}^{n} \frac{\mathrm{e}^{-\lambda r_i} (\lambda r_i)^{x_i}}{x_i!} p^{r_i} (1-p)^{1-r_i} .$$

We wish to sample from the posterior pdf $f(\lambda, p, \mathbf{r} \mid \mathbf{x})$ using the Gibbs sampler.

a) Show that

1. $(\lambda \mid p, \mathbf{r}, \mathbf{x}) \sim \mathsf{Gamma}(a + \sum_i x_i,\ b + \sum_i r_i)$.
2. $(p \mid \lambda, \mathbf{r}, \mathbf{x}) \sim \mathsf{Beta}(1 + \sum_i r_i,\ n + 1 - \sum_i r_i)$.
3. $(r_i \mid \lambda, p, \mathbf{x}) \sim \mathsf{Ber}\left(\frac{p \mathrm{e}^{-\lambda}}{p \mathrm{e}^{-\lambda} + (1-p) I_{\{x_i = 0\}}}\right)$.

b) Generate a random sample of size $n = 100$ for the ZIP model using parameters $p = 0.3$ and $\lambda = 2$.

c) Implement the Gibbs sampler, generate a large (dependent) sample from the posterior distribution and use this to construct 95% Bayesian CIs for p and λ using the data in b). Compare these with the true values.

6.15 * Show that μ in (6.15) satisfies the local balance equations

$$\mu(\mathbf{x}, \mathbf{y})\, \mathbf{R}[(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')] = \mu(\mathbf{x}', \mathbf{y}')\, \mathbf{R}[(\mathbf{x}', \mathbf{y}'), (\mathbf{x}, \mathbf{y})] .$$

Thus μ is stationary with respect to $\mathbf{R}$, that is, $\mu\mathbf{R} = \mu$. Show that μ is also stationary with respect to $\mathbf{Q}$. Show, finally, that μ is stationary with respect to $\mathbf{P} = \mathbf{QR}$.

6.16 * This is to show that the systematic Gibbs sampler is a special case of the generalized Markov sampler. Take $\mathscr{Y}$ to be the set of indices $\{1, \ldots, n\}$, and define for the Q-step

$$\mathbf{Q}_{\mathbf{x}}(y, y') = \begin{cases} 1 & \text{if } y' = y + 1 \text{ or } y' = 1, y = n \\ 0 & \text{otherwise.} \end{cases}$$

Let the set of possible transitions $\mathscr{R}(\mathbf{x}, y)$ be the set of vectors $\{(\mathbf{x}', y)\}$ such that all coordinates of $\mathbf{x}'$ are the same as those of $\mathbf{x}$ except for possibly the y-th coordinate.

a) Show that the stationary distribution of $\mathbf{Q}_{\mathbf{x}}$ is $q_{\mathbf{x}}(y) = 1/n$, for $y = 1, \ldots, n$.

b) Show that

$$\mathbf{R}[(\mathbf{x}, y), (\mathbf{x}', y)] = \frac{f(\mathbf{x}')}{\sum_{(\mathbf{z}, y) \in \mathscr{R}(\mathbf{x}, y)} f(\mathbf{z})}, \quad \text{for } (\mathbf{x}', y) \in \mathscr{R}(\mathbf{x}, y) .$$

c) Compare with Algorithm 6.4.1.

6.17 * Prove that the Metropolis–Hastings algorithm is a special case of the generalized Markov sampler. (Hint: let the auxiliary set $\mathscr{Y}$ be a copy of the target set $\mathscr{X}$, let $\mathbf{Q}_{\mathbf{x}}$ correspond to the transition function of the Metropolis–Hastings algorithm (that is, $\mathbf{Q}_{\mathbf{x}}(\cdot, \mathbf{y}) = q(\mathbf{x}, \mathbf{y})$), and define $\mathscr{R}(\mathbf{x}, \mathbf{y}) = \{(\mathbf{x}, \mathbf{y}), (\mathbf{y}, \mathbf{x})\}$. Use arguments similar to those for the Markov jump sampler (see (6.20)) to complete the proof.)

6.18 * Barker's and Hastings' MCMC algorithms differ from the symmetric Metropolis sampler only in that they define the acceptance ratio $\alpha(\mathbf{x}, \mathbf{y})$ to be respectively $f(\mathbf{y})/(f(\mathbf{x}) + f(\mathbf{y}))$ and $s(\mathbf{x}, \mathbf{y})/(1 + 1/\varrho(\mathbf{x}, \mathbf{y}))$ instead of $\min\{f(\mathbf{y})/f(\mathbf{x}), 1\}$. Here $\varrho(\mathbf{x}, \mathbf{y})$ is defined in (6.6) and s is any symmetric function such that $0 \leqslant \alpha(\mathbf{x}, \mathbf{y}) \leqslant 1$. Show that both are special cases of the generalized Markov sampler. (Hint: take $\mathscr{Y} = \mathscr{X}$.)

6.19 Implement the simulated annealing algorithm for the n-queens problem suggested in Example 6.13. How many solutions can you find?

6.20 Implement the Metropolis–Hastings based simulated annealing algorithm for the TSP in Example 6.12. Run the algorithm on some test problems in

`http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`

6.21 Write a simulated annealing algorithm based on the random walk sampler to maximize the function

$$S(x) = \left| \frac{\sin^8(10x) + \cos^5(5x + 1)}{x^2 - x + 1} \right|, \quad x \in \mathbb{R} .$$

Use a $\mathrm{N}(x, \sigma^2)$ proposal function, given the current state x. Start with $x = 0$. Plot the current best function value against the number of evaluations of S for various values of σ and various annealing schedules. Repeat the experiments several times to assess what works best.

Further Reading

MCMC is one of the principal tools of statistical computing and Bayesian analysis. A comprehensive discussion of MCMC techniques can be found in [19], and practical applications

are discussed in [7]. For more details on the use of MCMC in Bayesian analysis, we refer to [5]. A classical reference on simulated annealing is [1]. More general global search algorithms may be found in [25]. An influential paper on stationarity detection in Markov chains, which is closely related to perfect sampling, is [3].

REFERENCES

1. E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons, Chichester, 1989.
2. D. J. Aldous and J. Fill. *Reversible Markov Chains and Random Walks on Graphs.* In preparation. `http://www.stat.berkeley.edu /users/aldous/book.html`, 2007.
3. S. Asmussen, P. W. Glynn, and H. Thorisson. Stationary detection in the initial transient problem. *ACM Transactions on Modeling and Computer Simulation*, 2(2):130–157, 1992.
4. S. Baumert, A. Ghate, S Kiatsupaibul, Y. Shen, R. L. Smith, and Z. B. Zabinsky. A discrete hit-and-run algorithm for generating multivariate distributions over arbitrary finite subsets of a lattice. Technical report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, 2006.
5. A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis.* Chapman & Hall, New York, 2nd edition, 2003.
6. S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. *IEEE Transactions on PAMI*, 6:721–741, 1984.
7. W.R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice.* Chapman & Hall, New York, 1996.
8. P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995.
9. J. Hammersley and M. Clifford. Markov fields on finite graphs and lattices. Unpublished manuscript, 1970.
10. W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:92–109, 1970.
11. J. M. Keith, D. P. Kroese, and D. Bryant. A generalized Markov chain sampler. *Methodology and Computing in Applied Probability*, 6(1):29–53, 2004.
12. F. P. Kelly. *Reversibility and Stochastic Networks.* Wiley, Chichester, 1979.
13. J. S. Liu. *Monte Carlo Strategies in Scientifi c Computing.* Springer-Verlag, New York, 2001.
14. L. Lovász. Hit-and-run mixes fast. *Mathematical Programming*, 86:443–461, 1999.
15. L. Lovász and S. S. Vempala. Hit-and-run is fast and fun. Technical report, Microsoft Research, SMS-TR, 2003.
16. L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM Journal on Computing*, 35(4):985–1005, 2006.
17. M. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.
18. J. G. Propp and D. B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 1 & 2:223–252, 1996.
19. C. P. Robert and G. Casella. *Monte Carlo Statistical Methods.* Springer, New York, 2nd edition, 2004.

20. H. E. Romeijn and R. L. Smith. Simulated annealing for constrained global optimization. *Journal of Global Optimization*, 5:101–126, 1994.

21. S. M. Ross. *Simulation*. Academic Press, New York, 3rd edition, 2002.

22. Y. Shen. *Annealing Adaptive Search with Hit-and-Run Sampling Methods for Stochastic Global Optimization Algorithms*. PhD thesis, University of Washington, 2005.

23. Y. Shen, S. Kiatsupaibul, Z. B. Zabinsky, and R. L. Smith. An analytically derived cooling schedule for simulated annealing. *Journal of Global Optimization*, 38(3):333–365, 2007.

24. R. L. Smith. Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32:1296–1308, 1984.

25. Z. B. Zabinsky. *Stochastic Adaptive Search for Global Optimization*. Kluwer Academic Publishers, Dordrecht, 2003.

26. Z. B. Zabinsky, R. L. Smith, J. F. McDonald, H. E. Romeijn, and D. E. Kaufman. Improving hit-and-run for global optimization. *Journal of Global Optimization*, 3:171–192, 1993.

CHAPTER 7

SENSITIVITY ANALYSIS AND MONTE CARLO OPTIMIZATION

7.1 INTRODUCTION

As discussed in Chapter 3, many real-world complex systems in science and engineering can be modeled as *discrete-event systems*. The behavior of such systems is identified via a sequence of discrete events, which causes the system to change from one state to another. Examples include traffic systems, flexible manufacturing systems, computer-communications systems, inventory systems, production lines, coherent lifetime systems, PERT networks, and flow networks. A discrete-event system can be classified as either *static* or *dynamic*. The former are called *discrete-event static systems* (DESS), while the latter are called *discrete-event dynamic systems* (DEDS). The main difference is that DESS do not evolve over time, while DEDS do. The PERT network is a typical example of a DESS, with the sample performance being, for example, the shortest path in the network. A queueing network, such as the Jackson network in Section 3.3.1, is an example of a DEDS, with the sample performance being, for example, the delay (waiting time of a customer) in the network. In this chapter we shall deal mainly with DESS. For a comprehensive study of both DESS and DEDS the reader is referred to [11], [16], and [20].

Because of their complexity, the performance evaluation of discrete-event systems is usually studied by simulation, and it is often associated with the estimation of the performance or response function $\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})]$, where the distribution of the sample performance $H(\mathbf{X})$ depends on the control or reference parameter $\mathbf{u} \in \mathcal{V}$. *Sensitivity analysis* is concerned with evaluating sensitivities (gradients, Hessians, etc.) of the response function $\ell(\mathbf{u})$ with respect to parameter vector $\mathbf{u}$, and it is based on the score function and the Fisher infor-

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

mation. It provides guidance for design and operational decisions and plays an important role in selecting system parameters that optimize certain performance measures.

To illustrate, consider the following examples:

1. **Stochastic networks.** One might wish to employ sensitivity analysis in order to minimize the mean shortest path in the network with respect, say, to network link parameters, subject to certain constraints. PERT networks and flow networks are common examples. In the former, input and output variables may represent activity durations and minimum project duration, respectively. In the latter, they may represent flow capacities and maximal flow capacities.

2. **Traffic light systems.** Here the performance measure might be a vehicle's average delay as it proceeds from a given origin to a given destination or the average number of vehicles waiting for a green light at a given intersection. The sensitivity and decision parameters might be the average rate at which vehicles arrive at intersections and the rate of light changes from green to red. Some performance issues of interest are:

   (a) What will the vehicle's average delay be if the interarrival rate at a given intersection increases (decreases), say, by 10–50%? What would be the corresponding impact of adding one or more traffic lights to the system?

   (b) Which parameters are most significant in causing bottlenecks (high congestion in the system), and how can these bottlenecks be prevented or removed most effectively?

   (c) How can the average delay in the system be minimized, subject to certain constraints?

We shall distinguish between the so-called *distributional* sensitivity parameters and the *structural* ones. In the former case we are interested in sensitivities of the expected performance

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})] = \int H(\mathbf{x}) f(\mathbf{x}; \mathbf{u})\, d\mathbf{x} \tag{7.1}$$

with respect to the parameter vector $\mathbf{u}$ of the pdf $f(\mathbf{x}; \mathbf{u})$, while in the latter case we are interested in sensitivities of the expected performance

$$\ell(\mathbf{u}) = \mathbb{E}[H(\mathbf{X}; \mathbf{u})] = \int H(\mathbf{x}; \mathbf{u}) f(\mathbf{x})\, d\mathbf{x} \tag{7.2}$$

with respect to the parameter vector $\mathbf{u}$ in the sample performance $H(\mathbf{x}; \mathbf{u})$. As an example, consider a $GI/G/1$ queue. In the first case $\mathbf{u}$ might be the vector of the interarrival and service rates, while in the second case $\mathbf{u}$ might be the buffer size. Note that often the parameter vector $\mathbf{u}$ includes both the distributional and structural parameters. In such a case, we shall use the following notation:

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X}; \mathbf{u}_2)] = \int H(\mathbf{x}; \mathbf{u}_1) f(\mathbf{x}; \mathbf{u}_2)\, d\mathbf{x}\,, \tag{7.3}$$

where $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$. Note that $\ell(\mathbf{u})$ in (7.1) and (7.2) can be considered particular cases of $\ell(\mathbf{u})$ in (7.3), where the corresponding sizes of the vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ equal 0.

■ **EXAMPLE 7.1**

Let $H(\mathbf{X}; u_3, u_4) = \max\{X_1 + u_3, X_2 + u_4\}$, where $\mathbf{X} = (X_1, X_2)$ is a two-dimensional vector with independent components and $X_i \sim f_i(X; u_i)$, $i = 1, 2$. In this case u_1 and u_2 are distributional parameters, while u_3 and u_4 are structural ones.

Consider the following minimization problem using representation (7.3) :

$$
\begin{array}{lll}
\text{minimize} & \ell_0(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_0(\mathbf{X}; \mathbf{u}_2)], & \mathbf{u} \in \mathscr{Y}, \\
\text{subject to :} & \ell_j(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_j(\mathbf{X}; \mathbf{u}_2)] \leqslant 0, & j = 1, \ldots, k, \\
& \ell_j(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_j(\mathbf{X}; \mathbf{u}_2)] = 0, & j = k+1, \ldots, M,
\end{array}
\qquad (\mathrm{P}_0) \qquad (7.4)
$$

where $H_j(\mathbf{X})$ is the j-th sample performance, driven by an input vector $\mathbf{X} \in \mathbb{R}^n$ with pdf $f(\mathbf{x}; \mathbf{u}_1)$, and $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ is a decision parameter vector belonging to some parameter set $\mathscr{Y} \subset \mathbb{R}^m$.

When the objective function $\ell_0(\mathbf{u})$ and the constraint functions $\ell_j(\mathbf{u})$ are available analytically, (P_0) becomes a standard nonlinear programming problem, which can be solved either analytically or numerically by standard nonlinear programming techniques. For example, the Markovian queueing system optimization falls within this domain. Here, however, it will be assumed that the objective function and some of the constraint functions in (P_0) are not available analytically (typically due to the complexity of the underlying system), so that one must resort to stochastic optimization methods, particularly Monte Carlo optimization.

The rest of this chapter is organized as follows. Section 7.2 deals with sensitivity analysis of DESS with respect to the distributional parameters. Here we introduce the celebrated *score function* (SF) method. Section 7.3 deals with simulation-based optimization for programs of type (P_0) when the expected values $\mathbb{E}_{\mathbf{u}_1}[H_j(\mathbf{X}, \mathbf{u}_2)]$ are replaced by their corresponding sample means. The simulation-based version of (P_0) is called the *stochastic counterpart* of the original program (P_0). The main emphasis will be placed on the stochastic counterpart of the unconstrained program (P_0). Here we show how the stochastic counterpart method can approximate quite efficiently the true unknown optimal solution of the program (P_0) using a single simulation. Our results are based on [15, 17, 18], where theoretical foundations of the stochastic counterpart method are established. It is interesting to note that Geyer and Thompson [2] independently discovered the stochastic counterpart method in 1995. They used it to make statistical inference for a particular unconstrained setting of the general program (P_0). Section 7.4 presents an introduction to sensitivity analysis and simulation-based optimization of DEDS. Particular emphasis is placed on sensitivity analysis with respect to the distributional parameters of Markov chains using the dynamic version of the SF method. For a comprehensive study on sensitivity analysis and optimization of DEDS, including different types of queueing and inventory models, the reader is referred to [16].

7.2 THE SCORE FUNCTION METHOD FOR SENSITIVITY ANALYSIS OF DESS

In this section we introduce the celebrated *score function (SF) method* for sensitivity analysis of DESS. The goal of the SF method is to estimate the gradient and higher derivatives of $\ell(\mathbf{u})$ with respect to the distributional parameter vector $\mathbf{u}$, where the expected performance

is given (see (7.1)) by

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})] ,$$

with $\mathbf{X} \sim f(\mathbf{x}; \mathbf{u})$. As we shall see below, the SF approach permits the estimation of *all* sensitivities (gradients, Hessians, etc.) from a *single simulation run* (experiment) for a DESS with tens and quite often with hundreds of parameters. We closely follow [16].

Consider first the case where $\mathbf{u}$ is scalar (denoted therefore u instead of $\mathbf{u}$) and assume that the parameter set $\mathscr{Y}$ is an open interval on the real line. Suppose that for all $\mathbf{x}$ the pdf $f(\mathbf{x}; u)$ is continuously differentiable in u and that there exists an integrable function $h(\mathbf{x})$ such that

$$\left| H(\mathbf{x}) \frac{\mathrm{d}f(\mathbf{x}; u)}{\mathrm{d}u} \right| \leqslant h(\mathbf{x}) \tag{7.5}$$

for all $u \in \mathscr{Y}$. Then under mild conditions [18] the differentiation and expectation (integration) operators are interchangeable, so that differentiation of $\ell(u)$ yields

$$\begin{aligned} \frac{\mathrm{d}\ell(u)}{\mathrm{d}u} &= \frac{\mathrm{d}}{\mathrm{d}u} \int H(\mathbf{x}) f(\mathbf{x}; u) \mathrm{d}\mathbf{x} = \int H(\mathbf{x}) \frac{\mathrm{d}f(\mathbf{x}; u)}{\mathrm{d}u} \, \mathrm{d}\mathbf{x} \\ &= \int H(\mathbf{x}) \frac{\frac{\mathrm{d}f(\mathbf{x};u)}{\mathrm{d}u}}{f(\mathbf{x}; u)} f(\mathbf{x}; u) \, \mathrm{d}\mathbf{x} = \mathbb{E}_u \left[H(\mathbf{X}) \frac{\mathrm{d} \ln f(\mathbf{X}; u)}{\mathrm{d}u} \right] \\ &= \mathbb{E}_u \left[H(\mathbf{X}) \, \mathcal{S}(u; \mathbf{X}) \right] , \end{aligned}$$

where

$$\mathcal{S}(u; \mathbf{x}) = \frac{\mathrm{d} \ln f(\mathbf{x}; u)}{\mathrm{d}u}$$

is the *score function* (SF); see also (1.64). It is viewed as a function of u for a given $\mathbf{x}$.

Consider next the multidimensional case. Similar arguments allow us to represent the gradient and the higher-order derivatives of $\ell(\mathbf{u})$ in the form

$$\nabla^k \ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}} \left[H(\mathbf{X}) \, \mathcal{S}^{(k)}(\mathbf{u}; \mathbf{X}) \right] , \tag{7.6}$$

where

$$\mathcal{S}^{(k)}(\mathbf{u}; \mathbf{x}) = \frac{\nabla^k f(\mathbf{x}; \mathbf{u})}{f(\mathbf{x}; \mathbf{u})} \tag{7.7}$$

is the *k-th order score function*, $k = 0, 1, 2, \ldots$. In particular, $\mathcal{S}^{(0)}(\mathbf{u}; \mathbf{x}) = 1$ (by definition), $\mathcal{S}^{(1)}(\mathbf{u}; \mathbf{x}) = \mathcal{S}(\mathbf{u}; \mathbf{x}) = \nabla \ln f(\mathbf{x}; \mathbf{u})$ and $\mathcal{S}^{(2)}(\mathbf{u}; \mathbf{x})$ can be represented as

$$\begin{aligned} \mathcal{S}^{(2)}(\mathbf{u}; \mathbf{x}) &= \nabla \mathcal{S}(\mathbf{u}; \mathbf{x}) + \mathcal{S}(\mathbf{u}; \mathbf{x}) \, \mathcal{S}(\mathbf{u}; \mathbf{x})^T \\ &= \nabla^2 \ln f(\mathbf{x}; \mathbf{u}) + \nabla \ln f(\mathbf{x}; \mathbf{u}) \, \nabla \ln f(\mathbf{x}; \mathbf{u})^T , \end{aligned} \tag{7.8}$$

where $\nabla \ln f(\mathbf{x}; \mathbf{u})^T$ represents that transpose of the column vector $\nabla \ln f(\mathbf{x}; \mathbf{u})$ of partial derivatives of $\ln f(\mathbf{x}; \mathbf{u})$. Note that all partial derivatives are taken with respect to the components of the parameter vector $\mathbf{u}$.

Table 7.1 displays the score functions $\mathcal{S}(\mathbf{u}; x)$ calculated from (7.6) for the commonly used distributions given in Table A.1 in the Appendix. We take $\mathbf{u}$ to be the usual parameters for each distribution. For example, for the Gamma(α, λ) and N(μ, σ^2) distributions we take $\mathbf{u} = (\alpha, \lambda)$ and $\mathbf{u} = (\mu, \sigma)$, respectively.

Table 7.1 Score functions for commonly used distributions.

Distr.	$f(x;\mathbf{u})$	$\mathcal{S}(\mathbf{u};x)$
Exp(λ)	$\lambda \mathrm{e}^{-\lambda x}$	$\lambda^{-1} - x$
Gamma(α,λ)	$\dfrac{\lambda^{\alpha} x^{\alpha-1} \mathrm{e}^{-\lambda x}}{\Gamma(\alpha)}$	$\left(\ln(\lambda x) - \frac{\Gamma'(\alpha)}{\Gamma(\alpha)},\ \alpha\lambda^{-1} - x\right)$
N(μ,σ^2)	$\dfrac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$	$\left(\sigma^{-2}(x-\mu),\ -\sigma^{-1} + \sigma^{-3}(x-\mu)^2\right)$
Weib(α,λ)	$\alpha\lambda\,(\lambda x)^{\alpha-1}\,\mathrm{e}^{-(\lambda x)^{\alpha}}$	$\left(\alpha^{-1} + \ln(\lambda x)[1-(\lambda x)^{\alpha}],\ \frac{\alpha}{\lambda}[1-(\lambda x)^{\alpha}]\right)$
Bin(n,p)	$\binom{n}{x} p^x (1-p)^{n-x}$	$\dfrac{x-np}{p(1-p)}$
Poi(λ)	$\dfrac{\lambda^x \mathrm{e}^{-\lambda}}{x!}$	$\dfrac{x}{\lambda} - 1$
G(p)	$p(1-p)^{x-1}$	$\dfrac{1-p\,x}{p(1-p)}$

In general, the quantities $\nabla^k \ell(\mathbf{u})$, $k = 0, 1, \ldots$, are not available analytically, since the response $\ell(\mathbf{u})$ is not available. They can be estimated, however, via simulation as

$$\widehat{\nabla^k \ell(\mathbf{u})} = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i)\, \mathcal{S}^{(k)}(\mathbf{u};\mathbf{X}_i). \tag{7.9}$$

It is readily seen that the function $\ell(\mathbf{u})$ and *all* the sensitivities $\nabla^k \ell(\mathbf{u})$ can be estimated from a single simulation, since in (7.6) all of them are expressed as expectations with respect to the same pdf, $f(\mathbf{x};\mathbf{u})$.

The following two toy examples provide more details on the estimation of $\nabla \ell(\mathbf{u})$. Both examples are only for illustration, since $\nabla^k \ell(\mathbf{u})$ is available analytically.

■ **EXAMPLE 7.2**

Let $H(\mathbf{X}) = X$, with $X \sim \mathrm{Ber}(p = u)$, where $u \in [0,1]$. Using Table 7.1 for the Bin$(1,p)$ distribution, we find immediately that the estimator of $\nabla \ell(u)$ is

$$\widehat{\nabla \ell}(u) = \frac{1}{N} \sum_{i=1}^{N} X_i \frac{X_i - u}{u(1-u)} = \frac{1}{u\,N} \sum_{i=1}^{N} X_i \approx 1\,, \tag{7.10}$$

where $X_1, \ldots, X_N$ is a random sample from Ber(u). In the second equation we have used the fact that here $X_i^2 = X_i$. The approximation sign in (7.10) follows from the law of large numbers.

Suppose that $u = \frac{1}{2}$. Suppose also that we took a sample of size $N = 20$ from Ber$(\frac{1}{2})$ and obtained the following values:

$$\{x_1, \ldots, x_{20}\} = \{0,1,0,0,1,0,0,1,1,1,0,1,0,1,1,0,1,0,1,1\}.$$

From (7.10) we see that the sample derivative is $\widehat{\nabla \ell}(\frac{1}{2}) = 1.1$, while the true one is clearly $\nabla \ell(\frac{1}{2}) = 1$.

■ **EXAMPLE 7.3**

Let $H(\mathbf{X}) = X$, with $X \sim \mathsf{Exp}(\lambda = u)$. This is also a toy example since $\nabla \ell(u) = -1/u^2$. We see from Table 7.1 that $\mathcal{S}(u; x) = u^{-1} - x$, and therefore

$$\widehat{\nabla \ell}(u) = \frac{1}{N} \sum_{i=1}^{N} X_i \, (u^{-1} - X_i) \approx -\frac{1}{u^2} \tag{7.11}$$

is an estimator of $\nabla \ell(u)$, where $X_1, \ldots, X_N$ is a random sample from $\mathsf{Exp}(u)$.

■ **EXAMPLE 7.4 Example 7.1 (Continued)**

As before, let $H(\mathbf{X}; u_3, u_4) = \max\{X_1 + u_3, \; X_2 + u_4\}$, where $\mathbf{X} = (X_1, X_2)$ is a two-dimensional vector with independent components and $X_i \sim f_i(X, u_i)$, $i = 1, 2$. Suppose we are interested in estimating $\nabla \ell(\mathbf{u}_1)$ with respect to the distributional parameter vector $\mathbf{u}_1 = (u_1, u_2)$. We have

$$\widehat{\nabla \ell}(\mathbf{u}_1) = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i; u_3, u_4) \, \mathcal{S}(\mathbf{u}_1; \mathbf{X}_i) \, ,$$

where $\mathcal{S}(\mathbf{u}_1; \mathbf{X}_i)$ is the column vector $(\mathcal{S}(u_1; X_{1i}), \; \mathcal{S}(u_2; X_{2i}))^T$.

Next, we shall apply the importance sampling technique to estimate the sensitivities $\nabla^k \ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H(\mathbf{X}) \, \mathcal{S}^{(k)}(\mathbf{u}; \mathbf{X})]$ simultaneously for several values of $\mathbf{u}$. To this end, let $g(\mathbf{x})$ be the importance sampling density, assuming, as usual, that the support of $g(\mathbf{x})$ contains the support of $H(\mathbf{x}) f(\mathbf{x}; \mathbf{u})$ for all $\mathbf{u} \in \mathscr{V}$. Then $\nabla^k \ell(\mathbf{u})$ can be written as

$$\nabla^k \ell(\mathbf{u}) = \mathbb{E}_g[H(\mathbf{X}) \, \mathcal{S}^{(k)}(\mathbf{u}; \mathbf{X}) \, W(\mathbf{X}; \mathbf{u})] \, , \tag{7.12}$$

where

$$W(\mathbf{x}; \mathbf{u}) = \frac{f(\mathbf{x}; \mathbf{u})}{g(\mathbf{x})} \tag{7.13}$$

is the likelihood ratio of $f(\mathbf{x}; \mathbf{u})$ and $g(\mathbf{x})$. The likelihood ratio estimator of $\nabla^k \ell(\mathbf{u})$ can be written as

$$\widehat{\nabla^k \ell}(\mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i) \, \mathcal{S}^{(k)}(\mathbf{u}; \mathbf{X}_i) \, W(\mathbf{X}_i; \mathbf{u}) \, , \tag{7.14}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from $g(\mathbf{x})$. Note that $\widehat{\nabla^k \ell}(\mathbf{u})$ is an unbiased estimator of $\nabla^k \ell(\mathbf{u})$ for *all* $\mathbf{u}$. This means that by varying $\mathbf{u}$ and keeping g fixed we can, in principle, estimate unbiasedly the whole *response surface* $\{\nabla^k \ell(\mathbf{u}), \mathbf{u} \in \mathscr{V}\}$ from a *single simulation*. Often the importance sampling distribution is chosen in the *same* class of distributions as the original one. That is, $g(\mathbf{x}) = f(\mathbf{x}; \mathbf{v})$ for some $\mathbf{v} \in \mathscr{V}$. If not stated otherwise, we assume from now on that $g(\mathbf{x}) = f(\mathbf{x}; \mathbf{v})$, that is, we assume that the importance sampling pdf lies in the same parametric family as the original pdf $f(\mathbf{x}; \mathbf{u})$. If we denote the likelihood ratio estimator of $\ell(\mathbf{u})$ for a given $\mathbf{v}$ by $\widehat{\ell}(\mathbf{u}; \mathbf{v})$, that is,

$$\widehat{\ell}(\mathbf{u}; \mathbf{v}) = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{X}_i) \, W(\mathbf{X}_i; \mathbf{u}, \mathbf{v}) \, , \tag{7.15}$$

with $W(\mathbf{x};\mathbf{u},\mathbf{v}) = f(\mathbf{x};\mathbf{u})/f(\mathbf{x};\mathbf{v})$, and the estimators in (7.14) by $\widehat{\nabla^k \ell}(\mathbf{u};\mathbf{v})$, then (see Problem 7.4)

$$\widehat{\nabla^k \ell}(\mathbf{u};\mathbf{v}) = \nabla^k \widehat{\ell}(\mathbf{u};\mathbf{v}) = \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i)\, \mathcal{S}^{(k)}(\mathbf{u};\mathbf{X}_i)\, W(\mathbf{X}_i;\mathbf{u},\mathbf{v}) \,. \tag{7.16}$$

Thus, the estimators of sensitivities are simply the sensitivities of the estimators.

Next, we apply importance sampling to the two toy examples 7.2 and 7.3, and show how to estimate $\nabla^k \ell(\mathbf{u})$ simultaneously for different values of $\mathbf{u}$ using a single simulation from the importance sampling pdf $f(\mathbf{x};\mathbf{v})$.

■ **EXAMPLE 7.5 Example 7.2 (Continued)**

Consider again the Bernoulli toy example, with $H(\mathbf{X}) = X$ and $X \sim \mathsf{Ber}(u)$. Suppose that the importance sampling distribution is $\mathsf{Ber}(v)$, that is

$$g(x) = f(x;v) = v^x (1-v)^{1-x} \,, \quad x = 0, 1 \,.$$

Using importance sampling we can write $\nabla^k \ell(u)$ as

$$\nabla^k \ell(u) = \mathbb{E}_v \left[X \frac{u^X (1-u)^{1-X}}{v^X (1-v)^{1-X}} \mathcal{S}^{(k)}(u;X) \right],$$

where $X \sim \mathsf{Ber}(v)$. Recall that for $\mathsf{Bin}(1,u)$ we have $\mathcal{S}(u;x) = \frac{x-u}{u(1-u)}$. The corresponding likelihood ratio estimator of $\nabla^k \ell(u)$ is

$$\begin{aligned} \widehat{\nabla^k \ell}(u;v) &= \frac{1}{N}\sum_{i=1}^{N} X_i \frac{u^{X_i}(1-u)^{1-X_i}}{v^{X_i}(1-v)^{1-X_i}} \mathcal{S}^{(k)}(u;X_i) \\ &= \frac{u}{v}\frac{1}{N}\sum_{i=1}^{N} X_i\, \mathcal{S}^{(k)}(u;X_i) \,, \end{aligned} \tag{7.17}$$

where $X_1, \ldots, X_N$ is a random sample from $\mathsf{Ber}(v)$. In the second equation we have used the fact that X_i is either 0 or 1. For $k = 0$ we readily obtain

$$\widehat{\ell}(u;v) = \frac{u}{v}\frac{1}{N}\sum_{i=1}^{N} X_i \,,$$

which also follows directly from (7.15), and for $k = 1$ we have

$$\widehat{\nabla \ell}(u;v) = \frac{u}{v}\frac{1}{N}\sum_{i=1}^{N} X_i \frac{X_i - u}{u(1-u)} = \frac{1}{v}\frac{1}{N}\sum_{i=1}^{N} X_i \,, \tag{7.18}$$

which is the derivative of $\widehat{\ell}(u;v)$, as observed in (7.16). Note that in the special case where $v = u$, the likelihood ratio estimators $\widehat{\ell}(u;u)$ and $\widehat{\nabla \ell}(u;u)$ reduce to the CMC estimator $\frac{1}{N}\sum_{i=1}^{N} X_i$ (sample mean) and the earlier-derived score function estimator (7.10), respectively. As a simple illustration, suppose we took a sample from $\mathsf{Ber}(v = 1/2)$ of size $N = 20$ and obtained

$$\{x_1, \ldots, x_{20}\} = \{0,1,0,0,1,0,0,1,1,1,0,1,0,1,1,0,1,0,1,1\} \,.$$

Suppose that, using this sample, we wish to estimate the quantities $\ell(u) = \mathbb{E}_u[X]$ and $\nabla\ell(u)$ *simultaneously* for $u = 1/4$ and $u = 1/10$. We readily obtain

$$\widehat{\ell}(u = 1/4; v = 1/2) = \frac{1/4}{1/2}\frac{11}{20} = \frac{11}{40}\,,$$

$$\widehat{\ell}(u = 1/10; v = 1/2) = \frac{1/10}{1/2}\frac{11}{20} = \frac{11}{100}\,,$$

and $\widehat{\nabla\ell}(u; v) = 11/10$ for both $u = 1/4$ and $1/10$.

■ **EXAMPLE 7.6 Example 7.3 (Continued)**

Let us consider the estimation of $\nabla^k\ell(u)$ simultaneously for several values of u in the second toy example, namely, where $H(X) = X$ and $X \sim \text{Exp}(u)$. Selecting the importance sampling distribution as

$$g(x) = f(x; v) = v\,\mathrm{e}^{-vx},\ x > 0$$

for some $v > 0$ and using (7.14), we can express $\nabla^k\ell(u)$ as

$$\nabla^k\ell(u) = \mathbb{E}_v\left[X\frac{u\,\mathrm{e}^{-uX}}{v\,\mathrm{e}^{-vX}}\,\mathcal{S}^{(k)}(u; X)\right]\,,$$

where $X \sim \text{Exp}(v)$ and (see Table 7.1) $\mathcal{S}(u; x) = \frac{1-ux}{u}$. The sample average estimator of $\nabla^k\ell(u)$ (see (7.14)) is

$$\widehat{\nabla^k\ell}(u; v) = \frac{1}{N}\sum_{i=1}^{N} X_i\frac{u\,\mathrm{e}^{-uX_i}}{v\,\mathrm{e}^{-vX_i}}\,\mathcal{S}^{(k)}(u; X_i)\,, \tag{7.19}$$

where $X_1, \ldots, X_N$ is a random sample from $\text{Exp}(v)$. For $k = 0$ we have

$$\widehat{\ell}(u; v) = \frac{1}{N}\sum_{i=1}^{N} X_i\frac{u\,\mathrm{e}^{-uX_i}}{v\,\mathrm{e}^{-vX_i}} \approx \frac{1}{u}\,,$$

and for $k = 1$ we obtain

$$\widehat{\nabla\ell}(u; v) = \frac{1}{N}\sum_{i=1}^{N} X_i\frac{u\,\mathrm{e}^{-uX_i}}{v\,\mathrm{e}^{-vX_i}}\frac{1 - uX_i}{u} \approx -\frac{1}{u^2}\,, \tag{7.20}$$

which is the derivative of $\widehat{\ell}(u; v)$, as observed in (7.16). Note that in the particular case where $v = u$, the importance sampling estimators, $\widehat{\ell}(u; u)$ and $\widehat{\nabla\ell}(u; u)$, reduce to the sample mean (CMC estimator) and the SF estimator (7.11), respectively.

For a given importance sampling pdf $f(\mathbf{x}; \mathbf{v})$, the algorithm for estimating the sensitivities $\nabla^k\ell(\mathbf{u})$, $k = 0, 1, \ldots$, for *multiple values* $\mathbf{u}$ *from a single simulation run* is given next.

Algorithm 7.2.1

1. *Generate a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *from the importance sampling pdf* $f(\mathbf{x}; \mathbf{v})$, *which must be chosen in advance.*

2. *Calculate the sample performance* $H(\mathbf{X}_i)$ *and the scores* $\mathrm{S}^{(k)}(\mathbf{u}; \mathbf{X}_i)$, $i = 1, \ldots, N$, *for the desired parameter value(s)* $\mathbf{u}$.

3. *Calculate* $\widehat{\nabla^k \ell}(\mathbf{u}; \mathbf{v})$ *according to (7.16).*

From Algorithm 7.2.1 it follows that in order to estimate the sensitivities $\nabla^k \ell(\mathbf{u})$, $k = 1, 2, \ldots$, all we need is to apply formula (7.16), which involves calculation of the performance $H(\mathbf{X}_i)$ and estimation of the scores $\mathrm{S}^{(k)}(\mathbf{u}; \mathbf{X}_i)$ based on a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ obtained from the importance sampling pdf $f(\mathbf{x}; \mathbf{v})$.

Confidence regions for $\nabla^k \ell(\mathbf{u})$ can be obtained by standard statistical techniques. In particular (see, for example, [18] and Section 1.10), $N^{1/2}\left[\widehat{\nabla^k \ell}(\mathbf{u}; \mathbf{v}) - \nabla^k \ell(\mathbf{u})\right]$ converges to a multivariate normal random vector with mean zero and covariance matrix

$$\mathrm{Cov}_{\mathbf{v}}(H\,\mathrm{S}^{(k)}\,W) = \mathbb{E}_{\mathbf{v}}\left[H^2\,W^2\,\mathrm{S}^{(k)}\mathrm{S}^{(k)\,T}\right] - [\nabla^k \ell(\mathbf{u})][\nabla^k \ell(\mathbf{u})]^T, \tag{7.21}$$

using the abbreviations $H = H(\mathbf{X})$, $\mathrm{S}^{(k)} = \mathrm{S}^{(k)}(\mathbf{u}; \mathbf{x})$ and $W = W(\mathbf{X}; \mathbf{u}, \mathbf{v})$. From now on, we will use these abbreviations when convenient, abbreviating $\mathrm{S}^{(1)}$ further to S.

In particular, in the case $k = 0$, the variance of $\widehat{\ell}(\mathbf{u}; \mathbf{v})$, under the importance sampling density $f(\mathbf{x}; \mathbf{v})$, can be written as

$$\mathrm{Var}\left(\widehat{\ell}(\mathbf{u}; \mathbf{v})\right) = \mathbb{E}_{\mathbf{v}}\left[H^2\,W^2\right] - \ell^2(\mathbf{u})\,. \tag{7.22}$$

The crucial issue is, clearly, how to choose a good importance sampling pdf, which ensures low-variance estimates of $\ell(\mathbf{u})$ and $\nabla \ell(\mathbf{u})$. As we shall see, this is not a simple task. We start with the variance of $\widehat{\ell}(\mathbf{u}; \mathbf{v})$. We shall show that for exponential families of the form (A.9) it can be derived explicitly. Specifically, with $\boldsymbol{\theta}$ taking the role of $\mathbf{u}$ and $\boldsymbol{\eta}$ the role of $\mathbf{v}$, we have

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{\eta}}\left[H^2(\mathbf{X})W^2(\mathbf{X}; \boldsymbol{\theta}, \boldsymbol{\eta})\right] &= \mathbb{E}_{\boldsymbol{\theta}}\left[H^2(\mathbf{X})W(\mathbf{X}; \boldsymbol{\theta}, \boldsymbol{\eta})\right] \\
&= \int H^2(\mathbf{x}) \frac{c(\boldsymbol{\theta})}{c(\boldsymbol{\eta})}\, \mathrm{e}^{(\boldsymbol{\theta}-\boldsymbol{\eta})\cdot \mathbf{t}(\mathbf{x})}\, c(\boldsymbol{\theta})\, \mathrm{e}^{\boldsymbol{\theta}\cdot \mathbf{t}(\mathbf{x})}\, h(\mathbf{x})\, d\mathbf{x} \\
&= \frac{c^2(\boldsymbol{\theta})}{c(\boldsymbol{\eta})} \int H^2(\mathbf{x})\, \mathrm{e}^{(2\boldsymbol{\theta}-\boldsymbol{\eta})\cdot \mathbf{t}(\mathbf{x})}\, h(\mathbf{x})\, d\mathbf{x} \\
&= \frac{c^2(\boldsymbol{\theta})}{c(\boldsymbol{\eta})\, c(2\boldsymbol{\theta}-\boldsymbol{\eta})}\, \mathbb{E}_{2\boldsymbol{\theta}-\boldsymbol{\eta}}\left[H^2(\mathbf{X})\right] \\
&= \mathbb{E}_{\boldsymbol{\eta}}\left[W^2(\mathbf{X}; \boldsymbol{\theta}, \boldsymbol{\eta})\right]\; \mathbb{E}_{2\boldsymbol{\theta}-\boldsymbol{\eta}}\left[H^2(\mathbf{X})\right]\,.
\end{aligned} \tag{7.23}$$

Note that $\mathbb{E}_{\boldsymbol{\eta}}\left[W^2(\mathbf{X}; \boldsymbol{\theta}, \boldsymbol{\eta})\right] = \mathbb{E}_{\boldsymbol{\theta}}\left[W(\mathbf{X}; \boldsymbol{\theta}, \boldsymbol{\eta})\right]$.

Table 7.2 displays the expectation $\mathbb{E}_v\left[W^2(X; u, v)\right]$ for common exponential families in Tables A.1 and 7.1. Note that in Table 7.2 we change *one* parameter only, which is denoted by u and is changed to v. The values of $\mathbb{E}_v\left[W^2(X; u, v)\right]$ are calculated via (A.9)

and (7.23). In particular, we first reparameterize the distribution in terms of (A.9), with $\theta = \psi(u)$ and $\eta = \psi(v)$, and then calculate

$$\mathbb{E}_\eta \left[W^2(X; \theta, \eta) \right] = \frac{c^2(\theta)}{c(\eta)\, c(2\theta - \eta)} \,. \tag{7.24}$$

At the end, we substitute u and v back in order to obtain the desired $\mathbb{E}_v \left[W^2(X; u, v) \right]$.

Table 7.2 $\mathbb{E}_v[W^2]$ for commonly used distributions.

Distr.	$f(x;u)$	$\theta = \psi(u)$	$c(\theta)$	$\mathbb{E}_v[W^2(X;u,v)]$
Gamma(α, u)	$\dfrac{u^\alpha x^{\alpha-1} \mathrm{e}^{-ux}}{\Gamma(\alpha)}$	$-u$	$\dfrac{(-\theta)^\alpha}{\Gamma(\alpha)}$	$\left(\dfrac{u^2}{v(2u-v)}\right)^\alpha$
N(u, σ^2)	$\dfrac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{x-u}{\sigma}\right)^2}$	$\dfrac{u}{\sigma^2}$	$\dfrac{\mathrm{e}^{-\frac{1}{2}\theta^2\sigma^2}}{\sigma\sqrt{2\pi}}$	$\mathrm{e}^{\left(\frac{u-v}{\sigma}\right)^2}$
Weib(α, u)	$\alpha u\,(ux)^{\alpha-1} \mathrm{e}^{-(ux)^\alpha}$	$-u^\alpha$	$\alpha\theta$	$\dfrac{(u/v)^{2\alpha}}{2\,(u/v)^\alpha - 1}$
Bin(n, u)	$\binom{n}{x} u^x (1-u)^{n-x}$	$\ln\left(\dfrac{u}{1-u}\right)$	$(1+\mathrm{e}^\theta)^{-n}$	$\left(\dfrac{u^2 - 2uv + v}{(1-u)v}\right)^n$
Poi(u)	$\dfrac{u^x \mathrm{e}^{-u}}{x!}$	$\ln u$	$\mathrm{e}^{-\mathrm{e}^\theta}$	$\mathrm{e}^{\left(\frac{(u-v)^2}{v}\right)}$
G(u)	$u(1-u)^{x-1}$	$\ln(1-u)$	$1 - \mathrm{e}^\theta$	$\dfrac{u^2(v-1)}{v(u^2 - 2u + v)}$

Consider, for example, the Gamma(α, u) pdf. It readily follows that in order for the estimator $\widehat{\ell}(u; v)$ to be meaningful (Var$(\widehat{\ell}(u; v)) < \infty$), one should ensure that $2u - v > 0$, $(v < 2\,u)$; otherwise, W will "blow up" the variance of the importance sampling estimator $\widehat{\ell}(u; v)$. A more careful analysis [18] (see also Proposition A.4.2 in the Appendix) indicates that in this case v should be chosen smaller than u (instead of smaller than $2u$) because the optimal importance sampling pdf $f(x; v^*)$ has a "fatter" tail than the original pdf $f(x; u)$. A similar result holds for the estimators of $\nabla^k \ell(\mathbf{u})$ and for other exponential families.

Consider next the multidimensional case $\mathbf{X} = (X_1, \ldots, X_n)$. Assume for concreteness that the $\{X_i\}$ are independent and $X_i \sim \mathrm{Exp}(u_i)$. It is not difficult to derive (see Problem 7.3) that in this case

$$\mathbb{E}_{\mathbf{v}}[W^2] = \mathbb{E}_{\mathbf{u}}[W] = \prod_{k=1}^{n} \frac{1}{1 - \delta_k^2} \,, \tag{7.25}$$

where $\delta_k = (u_k - v_k)/u_k$, $k = 1, \ldots, n$ is the *relative perturbation* in u_k. For the special case where δ_k does not depend on k, say, $\delta_k = \delta$, $k = 1, \ldots, n$, we obtain

$$\mathrm{Var}_v(HW) = (1 - \delta^2)^{-n}\, \mathbb{E}_{2u-v} \left[H^2 \right] - \ell^2 \,. \tag{7.26}$$

We point out that for fixed δ (even with $v < 2u$, which corresponds to $\delta < 1$), the variance of HW increases exponentially in n. For small values of δ, the first term on the right-hand side of (7.26) can be approximated by

$$(1 - \delta^2)^{-n} = \exp\left\{ -n \ln\left(1 - \delta^2\right) \right\} \approx \exp\left\{ n\delta^2 \right\} \,,$$

using the fact that for small x, $\ln(1+x) \approx x$. This shows that in order for the variance of HW to be manageably small, the value $n\delta^2$ must not be too large. That is, as n increases, δ^2 should satisfy

$$\delta^2 = \mathcal{O}(n^{-1}) \,. \tag{7.27}$$

It is shown in [18] that an assumption similar to (7.27) must hold for rather general distributions and, in particular, for the exponential family.

Formula (7.27) is associated with the so-called *trust region*, that is, the region where the likelihood ratio estimator $\widehat{\nabla^k \ell}(\mathbf{u}; \mathbf{v})$ can be trusted to give a reasonably good approximation of $\nabla^k \ell(\mathbf{u})$. As an illustration, consider the case where $u_i = u$, $v_i = v$ for all i and $n = 100$. It can be seen that the estimator $\widehat{\ell}(u; v)$ performs reasonably well for δ not exceeding 0.1, that is, when the relative perturbation in u is within 10%. For larger relative perturbations, the term $\mathbb{E}_v[W^2]$ "blows up" the variance of the estimators. Similar results also hold for the derivatives of $\ell(u)$.

The above (negative) results on the unstable behavior of the likelihood ratio W and the rapid decrease of the trust region with the dimensionality n (see (7.27) do not leave much room for importance sampling to be used for estimation of $\nabla^k \ell(\mathbf{u})$, $k \geq 0$ in high dimensions. For such problems we suggest, therefore, the use of the score function estimators given in (7.9) (the ones that do not contain the likelihood ratio term W) as estimators of the true $\nabla^k \ell(\mathbf{u})$. For low-dimensional problems, say $n \leq 10$, one can still use the importance sampling estimator (7.14) for $\nabla^k \ell(\mathbf{u})$, provided that the trust region is properly chosen, say when the relative perturbation δ from the original parameter vector $\mathbf{u}$ does not exceed 10–20%. Even in this case, in order to prevent the degeneration of the importance sampling estimates, it is crucial to choose the reference parameter vector $\mathbf{v}$ such that the associated importance sampling pdf $f(\mathbf{x}; \mathbf{v})$ has a "fatter" tail than the original pdf $f(\mathbf{x}; \mathbf{u})$; see also Section A.4 of the Appendix.

7.3 SIMULATION-BASED OPTIMIZATION OF DESS

Consider the optimization program (P_0) in (7.4). Suppose that the objective function

$$\ell_0(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_0(\mathbf{X}; \mathbf{u}_2)]$$

and some of the constraint functions

$$\ell_j(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H_j(\mathbf{X}; \mathbf{u}_2)]$$

are not available in an analytical form, so that in order to solve (P_0) we must resort to simulation-based optimization, which involves using the sample average versions, $\widehat{\ell}_0(\mathbf{u})$ and $\widehat{\ell}_j(\mathbf{u})$ instead of $\ell_0(\mathbf{u})$ and $\ell_j(\mathbf{u})$, respectively. Recall that the parameter vector $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ can have distributional and structural components.

Next, we present a general treatment of simulation-based programs of type (P_0), with an emphasis on how to estimate the optimal solution $\mathbf{u}^*$ of the program (P_0) using a *single simulation* run. Assume that we are given a random sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ from the pdf $f(\mathbf{x}; \mathbf{u}_1)$ and consider the following two cases.

Case A. Either of the following holds true:

1. It is too expensive to store long samples $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ and the associated sequences $\{\widehat{\ell}_j(\mathbf{u})\}$.

2. The sample performance, $\widehat{\ell}_j(\mathbf{u})$, cannot be computed simultaneously for different values of $\mathbf{u}$. However, we are allowed to set the control vector, $\mathbf{u}$, at any desired value $\mathbf{u}^{(t)}$ and then compute the random variables $\widehat{\ell}_j(\mathbf{u}^{(t)})$ and (quite often) the associated derivatives (gradients) $\widehat{\nabla \ell_j}(\mathbf{u})$, at $\mathbf{u} = \mathbf{u}^{(t)}$.

Case B. Both of the following hold true:

1. It is easy to compute and store the whole sample, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$.

2. Given a sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$, it is easy to compute the sample performance $\widehat{\ell}_j(\mathbf{u})$ for any desired value $\mathbf{u}$.

From an application-oriented viewpoint, the main difference between Case *A* and Case *B* is that the former is associated with *on-line optimization*, also called *stochastic approximation*, while the latter with *off-line*, also called *stochastic counterpart optimization* or *sample average approximation*. As for references on stochastic approximation and the stochastic counterpart method we refer to [10] and [18], respectively.

The following two subsections deal separately with the stochastic approximation and the stochastic counterpart methods.

7.3.1 Stochastic Approximation

Stochastic approximation originated with the seminal papers of Robbins and Monro [13] and Kiefer and Wolfowitz [7]. The latter authors deal with on-line minimization of *smooth convex* problems of the form

$$\min_{\mathbf{u}} \ell(\mathbf{u}), \quad \mathbf{u} \in \mathscr{V}, \tag{7.28}$$

where it is assumed that the feasible set $\mathscr{V}$ is convex and that at any fixed-in-advance point $\mathbf{u} \in \mathscr{V}$ an estimate $\widehat{\nabla \ell}(\mathbf{u})$ of the true gradient $\nabla \ell(\mathbf{u})$ can be computed. Here we shall apply stochastic approximation in the context of simulation-based optimization.

The stochastic approximation method iterates in $\mathbf{u}$ using the following recursive formula:

$$\mathbf{u}^{(t+1)} = \Pi_{\mathscr{V}}(\mathbf{u}^{(t)} - \beta_t \widehat{\nabla \ell}(\mathbf{u}^{(t)})) , \tag{7.29}$$

where $\beta_1, \beta_2, \ldots$ is a sequence of positive step sizes and $\Pi_{\mathscr{V}}$ denotes the projection onto the set $\mathscr{V}$, that is, $\Pi_{\mathscr{V}}(\mathbf{u})$ is the point in $\mathscr{V}$ closest to $\mathbf{u}$. The projection $\Pi_{\mathscr{V}}$ is needed in order to enforce feasibility of the generated points $\{\mathbf{u}^{(t)}\}$. If the problem is unconstrained, that is, the feasible set $\mathscr{V}$ coincides with the whole space, then this projection is the identity mapping and can be omitted from (7.29).

It is readily seen that (7.29) represents a *gradient descent procedure* in which the exact gradients are replaced by their estimates. Indeed, if the exact value $\nabla \ell(\mathbf{u}^{(t)})$ of the gradient was available, then $-\nabla \ell(\mathbf{u}^{(t)})$ would give the direction of steepest descent at the point $\mathbf{u}^{(t)}$. This would guarantee that if $\nabla \ell(\mathbf{u}^{(t)}) \neq 0$, then moving along this direction the value of the objective function decreases, that is, $\ell(\mathbf{u}^{(t)} - \beta \nabla \ell(\mathbf{u}^{(t)})) < \ell(\mathbf{u}^{(t)})$ for $\beta > 0$ small enough. The iterative procedure (7.29) mimics that idea by using the estimates of the gradients instead of the true ones. Note again that a new random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ should be generated to calculate each $\widehat{\nabla \ell}(\mathbf{u}^{(t)})$, $t = 1, 2, \ldots$.

We shall now present several alternative estimators $\widehat{\nabla \ell}(\mathbf{u})$ of $\nabla \ell(\mathbf{u})$ considering the model in Example 7.1.

■ **EXAMPLE 7.7 Example 7.1 (Continued)**

As before, let $H(\mathbf{X}; u_3, u_4) = \max\{X_1 + u_3, X_2 + u_4\}$, where $\mathbf{X} = (X_1, X_2)$ is a two-dimensional vector with independent components, and $X_i \sim f_i(X; u_i)$, $i = 1, 2$. Assume that we are interested in estimating the four-dimensional vector $\nabla \ell(\mathbf{u})$, where $\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X}; \mathbf{u}_2)]$, $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) = (u_1, u_2, u_3, u_4)$, with respect to both the distributional parameter vector $\mathbf{u}_1 = (u_1, u_2)$, and the structural one $\mathbf{u}_2 = (u_3, u_4)$.

We shall now devise three alternative estimators for $\nabla \ell(\mathbf{u})$. They are called (a) the *direct*, (b) *inverse-transform*, and (c) *push-out* estimators. More details on these estimators and their various applications are given in [16].

(a) *The direct estimator of* $\nabla \ell(\mathbf{u})$. We have

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X}; \mathbf{u}_2)], \tag{7.30}$$

$$\frac{\partial \ell(\mathbf{u})}{\partial u_1} = \mathbb{E}_{\mathbf{u}_1}\left[H(\mathbf{X}; \mathbf{u}_2) \nabla \ln f_1(X_1; u_1)\right] , \tag{7.31}$$

$$\frac{\partial \ell(\mathbf{u})}{\partial u_3} = \mathbb{E}_{\mathbf{u}_1}\left[\frac{\partial H(\mathbf{X}; \mathbf{u}_2)}{\partial u_3}\right] , \tag{7.32}$$

and similarly for $\partial \ell(\mathbf{u})/\partial u_2$ and $\partial \ell(\mathbf{u})/\partial u_4$. Here

$$\frac{\partial H(\mathbf{X}; \mathbf{u}_2)}{\partial u_3} = \begin{cases} 1, & \text{if } X_1 + u_3 > X_2 + u_4 , \\ 0, & \text{otherwise,} \end{cases} \tag{7.33}$$

and similarly for $\partial H(\mathbf{X}; \mathbf{u}_2)/\partial u_4$. The sample estimators of $\partial \ell(\mathbf{u})/\partial u_i$, $i = 1, \ldots, 4$ can be obtained *directly* from their expected-value counterparts — hence the name *direct estimators*. For example, the estimator of $\partial \ell(\mathbf{u})/\partial u_3$ can be written as

$$\widehat{\nabla \ell}_3^{(1)}(\mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial H(\mathbf{X}_i; \mathbf{u}_2)}{\partial u_3} , \tag{7.34}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a sample from $f(\mathbf{x}; \mathbf{u}_1) = f_1(x_1; u_1)\, f_2(x_2; u_2)$, and similarly for the remaining estimators $\widehat{\nabla \ell}_i^{(1)}(\mathbf{u})$ of $\partial \ell(\mathbf{u})/\partial u_i$, $i = 1, 2, 4$.

(b) *The inverse-transform estimator of* $\nabla \ell(\mathbf{u})$. Using the inverse transformations $X_i = F_i^{-1}(Z_i; u_i)$, where $Z_i \sim \mathrm{U}(0, 1)$, $i = 1, 2$, we can write $H(\mathbf{X}; \mathbf{u}_2)$ alternatively as

$$\breve{H}(\mathbf{Z}; \mathbf{u}) = \max\{F_1^{-1}(Z_1; u_1) + u_3,\ F_2^{-1}(Z_2; u_2) + u_4\},$$

where $\mathbf{Z} = (Z_1, Z_1)$. The expected performance $\ell(\mathbf{u})$ and the gradient $\nabla \ell(\mathbf{u})$ can be now written as

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathrm{U}}[\breve{H}(\mathbf{Z}; \mathbf{u})]$$

and

$$\nabla \ell(\mathbf{u}) = \mathbb{E}_{\mathrm{U}}[\nabla \breve{H}(\mathbf{Z}; \mathbf{u})] ,$$

respectively. Here U denotes the uniform distribution. It is readily seen that in the inverse-transform setting all four parameters u_1, u_2, u_3, u_4 become *structural* ones. The estimator of $\nabla \ell(\mathbf{u})$ based on the inverse-transform method, denoted as $\widehat{\nabla \ell}^{(2)}(\mathbf{u})$, is therefore

$$\widehat{\nabla \ell}^{(2)}(\mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} \nabla \breve{H}(\mathbf{Z}_i; \mathbf{u}) , \tag{7.35}$$

where the partial derivatives of $\breve{H}(\mathbf{z};\mathbf{u})$ can be obtained similarly to (7.33). Note that the first-order derivatives of $H(\mathbf{z};\mathbf{u})$ are piecewise-continuous functions with discontinuities at points for which $F_1^{-1}(z_2;u_1)+u_2 = F_2^{-1}(z_2;u_2)+u_4$.

(c) *The push-out estimator of* $\nabla\ell(\mathbf{u})$. Define the following two random variables: $\widetilde{X}_1 = X_1 + u_3$ and $\widetilde{X}_2 = X_2 + u_4$. By doing so, the original sample performance $H(\mathbf{X};u_3,u_4) = \max\{X_1+u_3, X_2+u_4\}$ and the expected value $\ell(\mathbf{u})$ can be written as $\widetilde{H}(\widetilde{\mathbf{X}}) = \max\{\widetilde{X}_1,\widetilde{X}_2\}$ and as

$$\ell(\mathbf{u}) = \mathbb{E}_{\widetilde{f}}[\widetilde{H}(\widetilde{\mathbf{X}})] = \mathbb{E}_{\widetilde{f}}[\max\{\widetilde{X}_1,\widetilde{X}_2\}]\,, \tag{7.36}$$

respectively. Here $\widetilde{f}$ is the pdf of $\widetilde{\mathbf{X}}$; thus, $\widetilde{f}(\mathbf{x};\mathbf{u}) = f_1(x_1-u_3;u_1)\,f_2(x_2-u_4;u_2) = \widetilde{f}_1(x;u_1,u_3)\,\widetilde{f}_2(x;u_2,u_4)$. In this case we say that the original structural parameters u_3 and u_4 in $H(\cdot)$ are "pushed out" into the pdf $\widetilde{f}$.

As an example, suppose that $X_j \sim \mathsf{Exp}(u_j)$, $j=1,2$. Then the cdf $\widetilde{F}_1(x)$ of $\widetilde{X}_1 = X_1+u_3$ and the cdf $\widetilde{F}_2(x)$ of $\widetilde{X}_2 = X_2+u_4$ can be written, respectively, as

$$\widetilde{F}_1(x) = P(\widetilde{X}_1 \leqslant x) = \mathbb{P}(X_1 \leqslant x-u_3) = F_1(x-u_3)$$

and

$$\widetilde{F}_2(x) = \mathbb{P}(\widetilde{X}_2 \leqslant x-u_4) = F_2(x-u_4)\,.$$

Clearly,

$$\widetilde{f}_1(x;u_1,u_3) = \begin{cases} u_1\,\mathrm{e}^{-u_1(x-u_3)}, & x \geqslant u_3\,, \\ 0 & \text{otherwise,} \end{cases} \tag{7.37}$$

and

$$\widetilde{f}_2(x;u_2,u_4) = \begin{cases} u_2\,\mathrm{e}^{-u_2(x-u_4)}, & x \geqslant u_4 \\ 0 & \text{otherwise.} \end{cases} \tag{7.38}$$

It is readily seen that in the representation (7.36) all four parameters $u_1,\ldots,u_4$ are *distributional*. Thus, in order to estimate $\nabla^k\ell(\mathbf{u})$ we can apply the SF method. In particular, the gradient

$$\nabla\ell(\mathbf{u}) = \mathbb{E}_{\widetilde{f}}[\max\{\widetilde{X}_1,\widetilde{X}_2\}\,\nabla\ln\widetilde{f}(\widetilde{\mathbf{X}};\mathbf{u})]$$

can be estimated as

$$\widehat{\nabla\ell}^{(3)}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N}\max\{\widetilde{X}_{1i},\widetilde{X}_{2i}\}\nabla\ln\widetilde{f}(\widetilde{\mathbf{X}}_i;\mathbf{u})\,. \tag{7.39}$$

Recall that $\partial H(\mathbf{X};\mathbf{u}_1)/\partial u_1$ and $\partial H(\mathbf{X};u_1)/\partial u_2$ are piecewise-constant functions (see (7.33)) and that $\partial^2 H(\mathbf{X};\mathbf{u}_1)/\partial u_1^2$ and $\partial^2 H(\mathbf{X};\mathbf{u}_1)/\partial u_2^2$ vanish almost everywhere. Consequently, the associated second-order derivatives cannot be interchanged with the expectation operator in (7.30). On the other hand, the transformed function $\ell(\mathbf{u})$ in (7.36) and its sample version $\widehat{\nabla\ell}^{(3)}(\mathbf{u})$ are both *differentiable* in $\mathbf{u}$ everywhere, provided that $f_1(x_1-u_3;u_1)$ and $f_2(x_2-u_4;u_2)$ are smooth. Thus, subject to smoothness of $\widetilde{f}(\mathbf{x};\mathbf{u})$, the push-out technique *smoothes out* the original *non-smooth* performance measure $H(\cdot)$.

Let us return to stochastic optimization. As we shall see from Theorem 7.3.1 below, starting from some fixed initial value $\mathbf{u}^{(1)}$, under some reasonable assumptions the sequence

$\{\mathbf{u}^{(t)}\}$ converges asymptotically in t to the minimizer $\mathbf{u}^*$ of the objective function $\ell(\mathbf{u})$ over $\mathcal{V}$. Typically, in order to guarantee the convergence, the following two conditions are imposed on the step sizes: (a) $\sum_{t=1}^{\infty} \beta_t = \infty$, and (b) $\sum_{t=1}^{\infty} \beta_t^2 < \infty$. For example, one can take $\beta_t \equiv c/t$ for some $c > 0$. There are many theorems on the convergence and rate of convergence for stochastic optimization. One of the simplest, taken from [10], is presented next.

Theorem 7.3.1 *Assume that ℓ is smooth and strictly convex, that is,*

$$\ell(\mathbf{u}+\mathbf{h}) \geqslant \ell(\mathbf{u}) + [\nabla \ell(\mathbf{u})]^T \mathbf{h} + \frac{\beta}{2} \mathbf{h}^T \mathbf{h}, \quad \beta > 0 \,. \tag{7.40}$$

Assume further that the errors in the stochastic gradient vector $\widehat{\nabla \ell}(\mathbf{u})$ possess a bounded second moment, that is,

$$\mathbb{E}\left[\|\widehat{\nabla \ell}(\mathbf{u}) - \nabla \ell(\mathbf{u})\|^2\right] \leqslant C^2 < \infty \,.$$

Then, for an arbitrary (deterministic) positive sequence $\{\beta_t\}$ such that

$$\sum_{t=1}^{\infty} \beta_t = \infty, \quad \sum_{t=1}^{\infty} \beta_t^2 < \infty \,,$$

the vector $\mathbf{u}^{(t)}$ converges asymptotically to $\mathbf{u}^*$ (the minimizer of $\ell(\mathbf{u})$) in the sense of mean square. If, moreover,

$$\beta_t = c/t \,,$$

with an appropriate constant c (whether a given c is appropriate depends only on β in (7.40)), then for all t we have the following bounds:

$$\mathbb{E}\left[\|\mathbf{u}^{(t)} - \mathbf{u}^*\|^2\right] \leqslant \frac{A(\beta, c)}{t} \|\mathbf{u}^{(1)} - \mathbf{u}^*\|^2$$

and

$$\mathbb{E}\left[\ell(\mathbf{u}^{(t)}) - \ell(\mathbf{u}^*)\right] \leqslant \mathcal{O}(1/t) \,,$$

where $A(\beta, c)$ is some constant depending on β and c.

Attractive features of the stochastic approximation method are its simplicity and ease of implementation in those cases in which the projection $\Pi_{\mathcal{V}}(\cdot)$ can be easily computed. However, it also has severe shortcomings. The crucial question in implementations is the choice of the step sizes $\{\beta_t\}$. Small step sizes result in very slow progress toward the optimum, while large step sizes make the iterations "zigzag". Also, a few wrong steps in the beginning of the procedure may require many iterations to correct. For instance, the algorithm is extremely sensitive to the choice of the constant c in the step size rule $\beta_t = c/t$. Therefore, various step size rules have been suggested in which the step sizes are chosen adaptively. Another drawback of the stochastic approximation method is that it lacks good stopping criteria and often has difficulties handling even relatively simple linear constraints.

7.3.2 The Stochastic Counterpart Method

The underlying idea in the stochastic counterpart approach is to replace all the expected value functions in the deterministic program (P_0) by their sample average equivalents

and then solve the latter by standard mathematical programming techniques. The resultant optimal solution provides an estimator of the corresponding true optimal one for the original program (P_0).

If not stated otherwise, we shall consider here the unconstrained program

$$\min_{\mathbf{u}\in\mathscr{Y}} \ell(\mathbf{u}) = \min_{\mathbf{u}\in\mathscr{Y}} \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X};\mathbf{u}_2)] \,. \tag{7.41}$$

The general constrained program (P_0) is treated in [18].

Assume that $\mathscr{Y}$ is an open set and that $\ell(\mathbf{u})$ is continuously differentiable on $\mathscr{Y}$. Then, by the first-order necessary conditions, the gradient of $\ell(\mathbf{u})$ at an optimal solution point, $\mathbf{u}^*$, must vanish. Consequently, the optimal solution $\mathbf{u}^*$ can be found by solving the equation system

$$\nabla\ell(\mathbf{u}) = \mathbf{0} \,, \quad \mathbf{u}\in\mathscr{Y} \,. \tag{7.42}$$

Using the importance sampling pdf $f(\mathbf{x};\mathbf{v}_1)$, one can write the stochastic counterpart of (7.41) as

$$\min_{\mathbf{u}\in\mathscr{Y}} \widehat{\ell}(\mathbf{u};\mathbf{v}_1) = \min_{\mathbf{u}\in\mathscr{Y}} \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i;\mathbf{u}_2)\, W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1) \,, \tag{7.43}$$

where $\mathbf{X}_1,\ldots,\mathbf{X}_N$ is a random sample from the importance sampling pdf $f(\mathbf{x};\mathbf{v}_1)$ and

$$W(\mathbf{x};\mathbf{u}_1,\mathbf{v}_1) = \frac{f(\mathbf{x};\mathbf{u}_1)}{f(\mathbf{x};\mathbf{v}_1)} \,.$$

Assuming further that $\widehat{\ell}(\mathbf{u};\mathbf{v}_1)$ is continuously differentiable on $\mathscr{Y}$, the optimal solution of (7.43) can be estimated by solving the equation system

$$\widehat{\nabla\ell}(\mathbf{u};\mathbf{v}_1) = \nabla\widehat{\ell}(\mathbf{u};\mathbf{v}_1) = \mathbf{0} \,, \quad \mathbf{u}\in\mathscr{Y} \,, \tag{7.44}$$

which itself may be viewed as a stochastic counterpart of the deterministic system (7.42). Thus, we simply take the gradient of the likelihood ratio estimator $\widehat{\ell}(\mathbf{u};\mathbf{v}_1)$ as an estimator for the gradient of ℓ at $\mathbf{u}$; see also (7.16).

Note that (7.44) can be written as

$$\begin{aligned}\nabla\widehat{\ell}(\mathbf{u};\mathbf{v}_1) = \frac{1}{N}\sum_{i=1}^{N}\{&H(\mathbf{X}_i;\mathbf{u}_2)\nabla\ln f(\mathbf{X}_i;\mathbf{u}_1)\, W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1) \\ &+ \nabla H(\mathbf{X}_i;\mathbf{u}_2)\, W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1)\} = \mathbf{0}, \quad \mathbf{u}\in\mathscr{Y}.\end{aligned} \tag{7.45}$$

Recall that we can view the above problem as the sample average approximation of the true (or expected value) problem (7.41). The function $\widehat{\ell}(\mathbf{u};\mathbf{v}_1)$ is random in the sense that it depends on the corresponding sample $\mathbf{X}_1,\ldots,\mathbf{X}_N$. However, note that once the sample is generated, $\widehat{\ell}(\mathbf{u};\mathbf{v}_1)$ becomes a deterministic function whose values and derivatives can be computed for a given value of the argument $\mathbf{u}$. Consequently, the problem (7.43) becomes a deterministic optimization problem, and one can solve it with an appropriate deterministic optimization algorithm. For example, in the unconstrained case it can be solved by using, say, the steepest descent method, that is,

$$\mathbf{u}^{(t+1)} = \Pi_{\mathscr{Y}}(\mathbf{u}^{(t)} - \beta_t\nabla\widehat{\ell}(\mathbf{u}^{(t)};\mathbf{v}_1)) \,, \tag{7.46}$$

where the step size β_t is obtained by a line search, for example,

$$\beta_t \equiv \underset{\beta}{\operatorname{argmin}}\{\widehat{\ell}(\mathbf{u}^{(t)} - \beta\nabla\widehat{\ell}(\mathbf{u}^{(t)}; \mathbf{v}_1); \mathbf{v}_1)\} ,$$

and, as before, $\Pi_{\mathscr{Y}}$ denotes the projection onto the set $\mathscr{Y}$. Note that this procedure is different from the stochastic approximation method (7.29) in three respects:

1. The step sizes β_t are calculated by a line search instead of being defined a priori.
2. The same sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is used for all $\nabla\widehat{\ell}(\mathbf{u}^{(t)}; \mathbf{v}_1)$.
3. Typically, a reasonably large sample size N is used in (7.46) as compared to stochastic optimization in (7.29).

Next, we consider the particular case of the program (7.41) where $\mathbf{u}$ is a distributional decision parameter vector, that is, we consider the following program:

$$\min_{\mathbf{u}\in\mathscr{Y}} \ell(\mathbf{u}) = \min_{\mathbf{u}\in\mathscr{Y}} \mathbb{E}_{\mathbf{u}}[H(\mathbf{X})] . \tag{7.47}$$

To estimate the optimal solution $\mathbf{u}^*$ of the program (7.47), we shall use the score function method. In this case the program (7.43) reduces to

$$\min_{\mathbf{u}\in\mathscr{Y}} \widehat{\ell}(\mathbf{u}; \mathbf{v}) = \min_{\mathbf{u}\in\mathscr{Y}} \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i)\, W(\mathbf{X}_i; \mathbf{u}, \mathbf{v}) , \tag{7.48}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from the importance sampling pdf $f(\mathbf{x}; \mathbf{v})$, and

$$W(\mathbf{x}; \mathbf{u}, \mathbf{v}) = \frac{f(\mathbf{x}; \mathbf{u})}{f(\mathbf{x}; \mathbf{v})} .$$

By analogy to (7.45) we have

$$\frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i)\, \nabla \ln f(\mathbf{X}_i; \mathbf{u})\, W(\mathbf{X}_i; \mathbf{u}, \mathbf{v}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{Y}. \tag{7.49}$$

Remark 7.3.1 It is important to note that while solving the stochastic counterpart (7.48) based on likelihood ratios, the trust region issue becomes crucial. In particular, the following two requirements must hold:

(a) The reference parameter vector $\mathbf{v}$ must be chosen carefully, that is, the importance sampling pdf $f(\mathbf{x}; \mathbf{v})$ must have a "fatter" tail than the original pdf $f(\mathbf{x}; \mathbf{u})$ (see also Section A.4 of the appendix); otherwise, the degeneracy of the likelihood ratio $W(\mathbf{x}; \mathbf{u}, \mathbf{v})$ is likely to occur.

(b) The trust region $\mathscr{Y}$ of all possible values of $\mathbf{v}$ should be known in advance and $\mathscr{Y}$ should not be chosen too wide. In particular, it should satisfy (7.27). If the region $\mathscr{Y}$ is too wide, the likelihood ratio term $W(\mathbf{x}; \mathbf{u}, \mathbf{v})$ will "blow up" the variance of the estimate of (7.48). In this case, alternative methods (not involving likelihood ratio terms), like the steepest descent method (7.46), should be used. Common alternatives are the steepest descent and the stochastic approximation methods.

■ EXAMPLE 7.8 Stochastic Shortest Path

Consider the stochastic shortest path problem in Example 5.14 on page 140. Suppose the components $\{X_i\}$ are independent and $X_i \sim \mathsf{Exp}(u_i^{-1})$, $i = 1, \ldots, 5$, with $\mathbf{u} = (1, 2, u, 4, 5)$, for some $u > 0$. Suppose our objective is to solve the following program:

$$\min_{u \in \mathscr{V}} \ell(u) = \min_{u \in \mathscr{V}} [3.1 - \mathbb{E}_u[S(\mathbf{X})] + 0.1\, u] \,, \tag{7.50}$$

where $\mathscr{V} = \{u : 1 \le u \le 4\}$, $S(\mathbf{X}) = \min\{X_1 + X_4,\ X_1 + X_3 + X_5,\ X_2 + X_3 + X_4,\ X_2 + X_5\}$ denotes the length of the shortest path. The function $\ell(u)$ is difficult to evaluate exactly but can be estimated easily via Monte Carlo simulation, yielding a "noisy" function $\widehat{\ell}(u)$. Figure 7.1 displays estimates and confidence intervals for various values of u, using for each estimate a sample size of $N = 50{,}000$. The minimizer, say u^*, seems to lie in the interval [1,2].

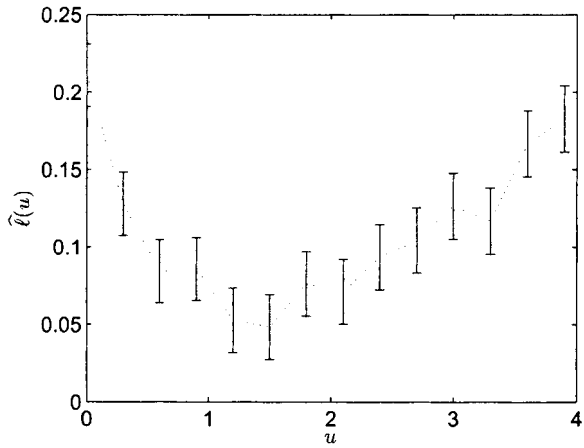


Figure 7.1 Minimize $\ell(u)$ with respect to u. Estimates and 95% confidence intervals indicate that the minimum is attained between 1 and 2.

To find u^* via the stochastic counterpart method, we can proceed as follows. First, the derivative of $\ell(u)$ satisfies

$$\begin{aligned} \nabla \ell(u) &= \frac{1}{10} - \mathbb{E}_v[S(\mathbf{X})\, \mathcal{S}(u; \mathbf{X})\, W(\mathbf{x}; u, v)] \\ &= \frac{1}{10} - \mathbb{E}_v\left[S(\mathbf{X}) \frac{X_3 - u}{u^2} \frac{v}{u} \mathrm{e}^{-X_3(u^{-1} - v^{-1})}\right] , \end{aligned}$$

where the score function $\mathcal{S}(u; \mathbf{X})$ is given by

$$\mathcal{S}(u; \mathbf{X}) = \frac{\partial}{\partial u} \ln\left(u^{-1} \mathrm{e}^{-X_3/u}\right) = \frac{X_3 - u}{u^2} \,. \tag{7.51}$$

Hence, the stochastic counterpart of $\nabla \ell(u) = 0$ is

$$\widehat{\nabla \ell}(u; v) = \frac{1}{10} - \frac{v}{N} \sum_{i=1}^{N} S(\mathbf{X}_i) \frac{X_{3i} - u}{u^3} \mathrm{e}^{-X_{3i}(u^{-1} - v^{-1})} = 0 \,,$$

with $\mathbf{X}_1, \ldots, \mathbf{X}_N$ simulated under parameter v. The choice of v here is *crucial*. In [16] the following method is proposed for choosing a "good" parameter v. First, one imposes some constraints on v, that is, $v_1 \leqslant v \leqslant v_2$, reflecting our prior knowledge about the optimal u^* (lying between v_1 and v_2). In our case, we could take $v_1 = 1$ and $v_2 = 4$, for example, since $\mathscr{V} = \{u :\leq u \leq 4\}$. Once these constraints have been determined, one takes v from this interval such that the tails of the corresponding distribution are as fat as possible to ensure that the likelihood ratio behaves well. In this case, it means taking $v = 4$.

Figure 7.2 shows the graph of $\widehat{\nabla \ell}(u; v)$ as a function of u (solid line). It was obtained from a sample of size $N = 500{,}000$ using $v = 4$. Recall that by definition $\nabla \ell(u^*) = 0$. As an estimate for u^* we take the point $\widehat{u^*}$ such that $\widehat{\nabla \ell}(\widehat{u^*}; v) = 0$. This can be found by standard root-finding procedures, for example, Matlab's `fzero` function. In this example, we find that $\widehat{u^*} = 1.37$.

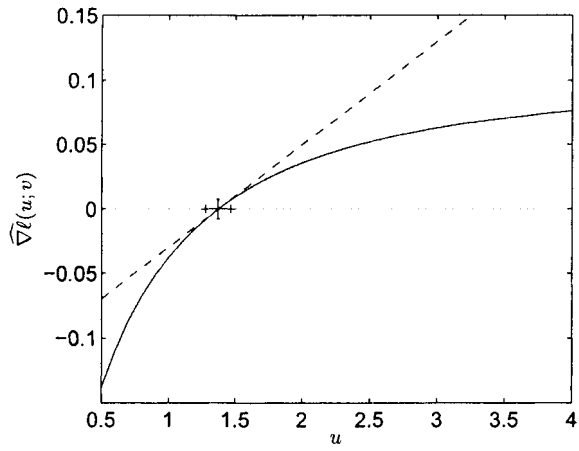


Figure 7.2 Estimate and 95% confidence interval for u^*. The estimate is found as the root of $\widehat{\nabla \ell}(u; v)$.

Next, we describe how to construct a confidence interval for u^* from a confidence interval for $\nabla \ell(u^*)$. For ease of notation we write $g(u)$ for $\nabla \ell(u)$ and $\widehat{g}(u)$ for its estimate, $\widehat{\nabla \ell}(u; v)$, and we assume that both $g(u)$ and $\widehat{g}(u)$ are monotonically increasing. Since $\widehat{g}(u)$ is of the form

$$\widehat{g}(u) = \frac{1}{10} - \frac{1}{N} \sum_{i=1}^{N} Z_i \,,$$

where $\{Z_i\}$ are iid random variables, an approximate $(1 - \alpha)$ confidence interval for $g(u^*) = 0$ is given by $(-C, C)$, with $C = z_{1-\alpha/2}\, S_Z / \sqrt{N}$. Here S_Z is the sample standard deviation of the $\{Z_i\}$ and $z_{1-\alpha/2}$ is the $1 - \alpha/2$ quantile of the standard normal distribution. As a consequence, $(g^{-1}(-C), g^{-1}(C))$ is an approximate $(1 - \alpha)$ confidence interval for u^*. For small C we have $g^{-1}(C) \approx u^* + C/g'(u^*)$, where

g' is the derivative of g, that is, the second derivative of ℓ. The latter is given by

$$g'(u) = \nabla^2 \ell(u) = -\mathbb{E}_v \left[S(\mathbf{X}) \frac{2u^2 - 4uX_3 + X_3^2}{u^4} \frac{v}{u} \mathrm{e}^{-X_3(u^{-1}-v^{-1})} \right]$$

and can be estimated via its stochastic counterpart, using the same sample as used to obtain $\widehat{g}(u)$. Indeed, the estimate of $g'(u)$ is simply the derivative of $\widehat{g}$ at u. Thus, an approximate $(1-\alpha)$ confidence interval for u^* is $\widehat{u^*} \pm C/\widehat{g}'(\widehat{u^*})$. This is illustrated in Figure 7.2, where the dashed line corresponds to the tangent line to $\widehat{g}(u)$ at the point $(\widehat{u}, 0)$, and 95% confidence intervals for $g(\widehat{u})$ and u^* are plotted vertically and horizontally, respectively. The particular values for these confidence intervals were found to be $(-0.0075, 0.0075)$ and $(1.28, 1.46)$.

Finally, it is important to choose the parameter v under which the simulation is carried out greater than u^*. This is highlighted in Figure 7.3, where 10 replications of $\widehat{g}(u)$ are plotted for the cases $v = 0.5$ and $v = 4$.

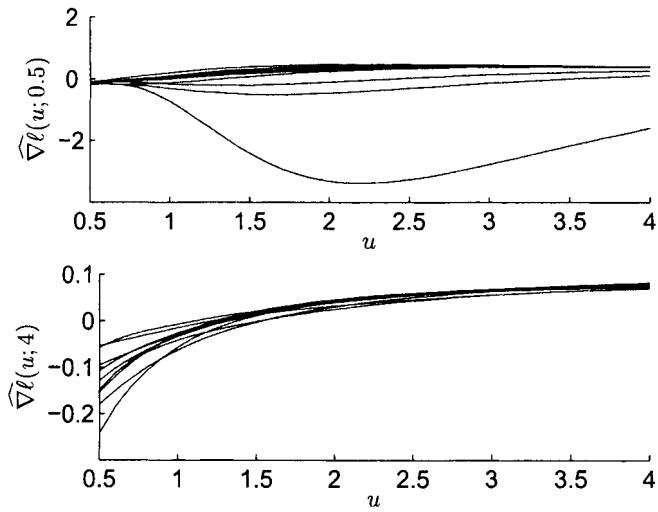


Figure 7.3 Ten replications of $\widehat{\nabla}\ell(u;v)$ are simulated under $v = 0.5$ and $v = 4$.

In the first case the estimates of $g(u) = \widehat{\nabla}\ell(u;v)$ fluctuate widely, whereas in the second case they remain stable. As a consequence, u^* cannot be reliably estimated under $v = 0.5$. For $v = 4$ no such problems occur. Note that this is in accordance with the general principle that the importance sampling distribution should have heavier tails than the target distribution. Specifically, under $v = 4$ the pdf of X_3 has heavier tails than under $v = u^*$, whereas the opposite is true for $v = 0.5$.

In general, let $\widehat{\ell}^*$ and $\widehat{\mathbf{u}}^*$ denote the optimal objective value and the optimal solution of the sample average problem (7.48), respectively. By the law of large numbers, $\widehat{\ell}(\mathbf{u};\mathbf{v})$ converges to $\ell(\mathbf{u})$ with probability 1 (w.p.1) as $N \to \infty$. One can show [18] that under mild additional conditions, $\widehat{\ell}^*$ and $\widehat{\mathbf{u}}^*$ converge w.p.1 to their corresponding optimal objective value and to the optimal solution of the true problem (7.47), respectively. That is, $\widehat{\ell}^*$ and $\widehat{\mathbf{u}}^*$

are consistent estimators of their true counterparts ℓ^* and $\mathbf{u}^*$, respectively. Moreover, [18] establishes a central limit theorem and valid confidence regions for the tuple $(\ell^*, \mathbf{u}^*)$. The following theorem summarizes the basic statistical properties of $\widehat{\mathbf{u}^*}$ for the unconstrained program formulation. Additional discussion, including proofs for both the unconstrained and constrained programs, may be found in [18].

Theorem 7.3.2 *Let $\mathbf{u}^*$ be a unique minimizer of $\ell(\mathbf{u})$ over $\mathcal{V}$.*

A. Suppose that

1. *The set $\mathcal{V}$ is compact.*
2. *For almost every $\mathbf{x}$, the function $f(\mathbf{x}; \cdot)$ is continuous on $\mathcal{V}$.*
3. *The family of functions $\{|H(\mathbf{x})f(\mathbf{x}; \mathbf{u})|, \mathbf{u} \in \mathcal{V}\}$ is dominated by an integrable function $h(\mathbf{x})$, that is,*

$$|H(\mathbf{x})f(\mathbf{x}; \mathbf{u})| \leqslant h(\mathbf{x}) \quad \text{for all } \mathbf{u} \in \mathcal{V}.$$

Then the optimal solution $\widehat{\mathbf{u}^*}$ of (7.48) converges to $\mathbf{u}^*$ as $N \to \infty$, with probability one.

B. Suppose further that

1. *$\mathbf{u}^*$ is an interior point of $\mathcal{V}$.*
2. *For almost every $\mathbf{x}$, $f(\mathbf{x}; \cdot)$ is twice continuously differentiable in a neighborhood $\mathcal{U}$ of $\mathbf{u}^*$, and the families of functions $\{\|H(\mathbf{x})\nabla^k f(\mathbf{x}; \mathbf{u})\| : \mathbf{u} \in \mathcal{U}, k = 1, 2\}$, where $\|\mathbf{x}\| = (x_1^2 + \cdots + x_n^2)^{\frac{1}{2}}$, are dominated by an integrable function.*
3. *The matrix*

$$B = \mathbb{E}_{\mathbf{v}}\left[H(\mathbf{X})\nabla^2 W(\mathbf{X}; \mathbf{u}^*, \mathbf{v})\right] \tag{7.52}$$

   is nonsingular.
4. *The covariance matrix of the vector $H(\mathbf{X})\nabla W(\mathbf{X}; \mathbf{u}^*, \mathbf{v})$, given by*

$$\Sigma = \mathbb{E}_{\mathbf{v}}\left[H^2(\mathbf{X})\nabla W(\mathbf{X}; \mathbf{u}^*, \mathbf{v})(\nabla W(\mathbf{X}; \mathbf{u}^*, \mathbf{v}))^T\right] - \nabla\ell(\mathbf{u}^*)(\nabla\ell(\mathbf{u}^*))^T,$$

   exists.

Then the random vector $N^{1/2}(\widehat{\mathbf{u}^*} - \mathbf{u}^*)$ converges in distribution to a normal random vector with zero mean and covariance matrix

$$B^{-1}\,\Sigma\,B^{-1}. \tag{7.53}$$

The asymptotic efficiency of the estimator $N^{1/2}(\widehat{\mathbf{u}^*} - \mathbf{u}^*)$ is controlled by the covariance matrix given in (7.53). Under the assumptions of Theorem 7.3.2, this covariance matrix can be consistently estimated by $\widehat{B}^{-1}\widehat{\Sigma}\widehat{B}^{-1}$, where

$$\widehat{B} = \frac{1}{N}\sum_{i=1}^{N} H(\mathbf{X}_i)\nabla^2 W(\mathbf{X}_i; \widehat{\mathbf{u}^*}, \mathbf{v}) \tag{7.54}$$

and

$$\widehat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N} H^2(\mathbf{X}_i)\,\nabla W(\mathbf{X}_i; \widehat{\mathbf{u}}, \mathbf{v})(\nabla W(\mathbf{X}_i; \widehat{\mathbf{u}^*}, \mathbf{v}))^T - \widehat{\nabla\ell}(\widehat{\mathbf{u}^*}; \mathbf{v})(\widehat{\nabla\ell}(\widehat{\mathbf{u}^*}; \mathbf{v}))^T \tag{7.55}$$

are consistent estimators of the matrices B and Σ, respectively. Observe that these matrices can be estimated from the same sample $\{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ simultaneously with the estimator $\widehat{\mathbf{u}^*}$. Observe also that the matrix B coincides with the Hessian matrix $\nabla^2 \ell(\mathbf{u}^*)$ and is, therefore, independent of the choice of the importance sampling parameter vector $\mathbf{v}$.

Although the above theorem was formulated for the distributional case only, similar arguments [18] apply to the stochastic counterpart (7.43), involving both distributional and structural parameter vectors $\mathbf{u}_1$ and $\mathbf{u}_2$, respectively.

The statistical inference for the estimators $\widehat{\ell^*}$ and $\widehat{\mathbf{u}^*}$ allows the construction of stopping rules, validation analysis and error bounds for the obtained solutions. In particular, it is shown in Shapiro [19] that if the function $\ell(\mathbf{u})$ is twice differentiable, then the above stochastic counterpart method produces estimators that converge to an optimal solution of the true problem at the same asymptotic rate as the stochastic approximation method, provided that the stochastic approximation method is applied with the *asymptotically optimal* step sizes. Moreover, it is shown in Kleywegt, Shapiro, and Homem de Mello [9] that if the underlying probability distribution is discrete and $\ell(\mathbf{u})$ is piecewise linear and convex, then w.p.1 the stochastic counterpart method (also called the *sample path method*) provides an exact optimal solution. For a recent survey on simulation-based optimization see Kleywegt and Shapiro [8].

The following example deals with unconstrained minimization of $\ell(\mathbf{u})$, where $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$ and therefore contains both distributional and structural parameter vectors.

■ **EXAMPLE 7.9 Examples 7.1 and 7.7 (Continued)**

Consider minimization of the function

$$\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}_1}[H(\mathbf{X}; \mathbf{u}_2)] + \mathbf{b}^T \mathbf{u} ,$$

where

$$H(\mathbf{X}; u_3, u_4) = \max\{X_1 + u_3, X_2 + u_4\} , \tag{7.56}$$

$\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2)$, $\mathbf{u}_1 = (u_1, u_2)$, $\mathbf{u}_2 = (u_3, u_4)$, $\mathbf{X} = (X_1, X_2)$ is a two-dimensional vector with independent components, $X_i \sim f_i(x; u_i)$, $i = 1, 2$, with $X_i \sim \mathsf{Exp}(u_i)$, and $\mathbf{b} = (b_1, \ldots, b_4)$ is a cost vector.

To find the estimate of the optimal solution $\mathbf{u}^*$ we shall use, by analogy to Example 7.7, the direct, inverse-transform and push-out estimators of $\nabla \ell(\mathbf{u})$. In particular, we shall define a system of nonlinear equations of type (7.44), which is generated by the corresponding direct, inverse-transform, and push-out estimators of $\nabla \ell(\mathbf{u})$. Note that each such estimator will be associated with a proper likelihood ratio function $W(\cdot)$.

(a) *The direct estimator of* $\nabla \ell(\mathbf{u})$. In this case

$$W(\mathbf{X}; \mathbf{u}_1, \mathbf{v}_1) = \frac{f_1(X_1; u_1) f_2(X_2; u_2)}{f_1(X_1; v_1) f_2(X_2; v_2)},$$

where $\mathbf{X} \sim f_1(x_1; v_1) f_2(x_2; v_2)$ and $\mathbf{v}_1 = (v_1, v_2)$. Using the above likelihood ratio term, formulas (7.31) and (7.32) can be written as

$$\frac{\partial \ell(\mathbf{u})}{\partial u_1} = \mathbb{E}_{\mathbf{v}_1}\left[H(\mathbf{X}; \mathbf{u}_2) W(\mathbf{X}; \mathbf{u}_1, \mathbf{v}_1) \nabla \ln f_1(X_1; u_1)\right] + b_1 \tag{7.57}$$

and as

$$\frac{\partial \ell(\mathbf{u})}{\partial u_3} = \mathbb{E}_{\mathbf{v}_1}\left[\frac{\partial H(\mathbf{X}; \mathbf{u}_2)}{\partial u_3} W(\mathbf{X}; \mathbf{u}_1, \mathbf{v}_1)\right] + b_3 , \tag{7.58}$$

respectively, and similarly $\partial \ell(\mathbf{u})/\partial u_2$ and $\partial \ell(\mathbf{u})/\partial u_4$. By analogy to (7.34) the importance sampling estimator of $\partial \ell(\mathbf{u})/\partial u_3$ can be written as

$$\widehat{\nabla \ell}_3^{(1)}(\mathbf{u};\mathbf{v}_1) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial H(\mathbf{X}_i;\mathbf{u}_2)}{\partial u_3} W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1) + b_3 \,, \tag{7.59}$$

where $\mathbf{X}_1,\ldots,\mathbf{X}_N$ is a random sample from $f(\mathbf{x};\mathbf{v}_1) = f_1(x_1;v_1)\, f_2(x_2;v_2)$, and similarly for the remaining importance sampling estimators $\widehat{\nabla \ell}_i^{(1)}(\mathbf{u};\mathbf{v}_1)$ of $\partial \ell(\mathbf{u})/\partial u_i$, $i = 1,2,4$. With this at hand, the estimate of the optimal solution $\mathbf{u}^*$ can be obtained from the solution of the following four-dimensional system of nonlinear equations:

$$\widehat{\nabla \ell}^{(1)}(\mathbf{u}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{Y}, \tag{7.60}$$

where $\widehat{\nabla \ell}^{(1)} = (\widehat{\nabla \ell}_1^{(1)},\ldots,\widehat{\nabla \ell}_4^{(1)})$.

(b) *The inverse-transform estimator of* $\nabla \ell(\mathbf{u})$. Taking (7.35) into account, the estimate of the optimal solution $\mathbf{u}^*$ can be obtained by solving, by analogy to (7.60), the following four-dimensional system of nonlinear equations:

$$\widehat{\nabla \ell}^{(2)}(\mathbf{u}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{Y} \,. \tag{7.61}$$

Here, as before,

$$\widehat{\nabla \ell}^{(2)}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} \nabla \breve{H}(\mathbf{Z}_i;\mathbf{u}) + \mathbf{b} \,,$$

and $\mathbf{Z}_1,\ldots,\mathbf{Z}_N$ is a random sample from the two-dimensional uniform pdf with independent components, that is, $\mathbf{Z} = (Z_1, Z_2)$ and $Z_j \sim \mathrm{U}(0,1)$, $j = 1,2$. Alternatively, one can estimate $\mathbf{u}^*$ using the ITLR method. In this case, by analogy to (7.61), the four-dimensional system of nonlinear equations can be written as

$$\widetilde{\nabla \ell}^{(2)}(\mathbf{u}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{Y} \tag{7.62}$$

with

$$\widetilde{\nabla \ell}^{(2)}(\mathbf{u}) = \frac{1}{N}\sum_{i=1}^{N} \nabla \breve{H}(\mathbf{X}_i;\mathbf{u}) W(\mathbf{X}_i;\boldsymbol{\theta}) + \mathbf{b} \,,$$

and

$$W(\mathbf{X}_i;\boldsymbol{\theta}) = \frac{1}{h_1(X_{1i};\theta_1)\, h_2(X_{2i};\theta_2)},$$

where $\boldsymbol{\theta} = (\theta_1,\theta_2)$, $\mathbf{X} = (X_1, X_2) \sim h_1(x_1;\theta_1) h_2(x_2;\theta_2)$ and, for example, $h_i(x;\theta_i) = \theta_i x^{\theta_i - 1}$, $i = 1,2$, that is, $h_i(\cdot)$ is a Beta pdf.

(c) *The push-out estimator of* $\nabla \ell(\mathbf{u})$. Taking (7.39) into account, the estimate of the optimal solution $\mathbf{u}^*$ can be obtained from the solution of the following four-dimensional system of nonlinear equations:

$$\widehat{\nabla \ell}^{(3)}(\mathbf{u};\mathbf{v}) = \mathbf{0}, \quad \mathbf{u} \in \mathscr{Y}, \tag{7.63}$$

where

$$\widehat{\nabla \ell}^{(3)}(\mathbf{u};\mathbf{v}) = \frac{1}{N}\sum_{i=1}^{N} \max\{\widetilde{X}_{1i}, \widetilde{X}_{2i}\}\, W(\widetilde{\mathbf{X}}_i;\mathbf{u},\mathbf{v})\nabla \ln \widetilde{f}(\widetilde{\mathbf{X}}_i;\mathbf{u}) + \mathbf{b}\,,$$

$$W(\widetilde{\mathbf{X}}_i;\mathbf{u},\mathbf{v}) = \frac{f_1(X_1 - u_3;u_1)\, f_2(X_2 - u_4;u_2)}{f_1(X_1 - v_3;v_1)\, f_2(X_2 - v_4;v_2)}$$

and $\widetilde{\mathbf{X}}_i \sim \widetilde{f}(\mathbf{x}) = f_1(x_1 - v_3;v_1)\, f_2(x_2 - v_4;v_2)$.

Let us return finally to the stochastic counterpart of the general program (P_0). From the foregoing discussion, it follows that it can be written as

$$(\widehat{\mathrm{P}}_N) \qquad \begin{array}{lll} \text{minimize} & \widehat{\ell}_0(\mathbf{u};\mathbf{v}_1), & \mathbf{u} \in \mathscr{V}, \\ \text{subject to:} & \widehat{\ell}_j(\mathbf{u};\mathbf{v}_1) \leqslant 0, & j = 1,\ldots,k, \\ & \widehat{\ell}_j(\mathbf{u};\mathbf{v}_1) = 0, & j = k+1,\ldots,M, \end{array} \tag{7.64}$$

with

$$\widehat{\ell}_j(\mathbf{u};\mathbf{v}_1) = \frac{1}{N}\sum_{i=1}^{N} H_j(\mathbf{X}_i;\mathbf{u}_2)\, W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1), \quad j = 0,1,\ldots,M, \tag{7.65}$$

where $\mathbf{X}_1,\ldots,\mathbf{X}_N$ is a random sample from the importance sampling pdf $f(\mathbf{x};\mathbf{v}_1)$, and the $\{\widehat{\ell}_j(\mathbf{u};\mathbf{v}_1)\}$ are viewed as functions of $\mathbf{u}$ rather than as estimators for a fixed $\mathbf{u}$.

Note again that once the sample $\mathbf{X}_1,\ldots,\mathbf{X}_N$ is generated, the functions $\widehat{\ell}_j(\mathbf{u};\mathbf{v}_1)$, $j = 0,\ldots,M$ become *explicitly* determined via the functions $H_j(\mathbf{X}_i;\mathbf{u}_2)$ and $W(\mathbf{X}_i;\mathbf{u}_1,\mathbf{v}_1)$. Assuming, furthermore, that the corresponding gradients $\nabla\widehat{\ell}_j(\mathbf{u};\mathbf{v}_1)$ can be calculated, for any $\mathbf{u}$, from a single simulation run, one can solve the optimization problem $(\widehat{\mathrm{P}}_N)$ by standard methods of mathematical programming. The resultant optimal function value and the optimal decision vector of the program $(\widehat{\mathrm{P}}_N)$ provide estimators of the optimal values ℓ^* and $\mathbf{u}^*$, respectively, of the original one (P_0). It is important to understand that what makes this approach feasible is the fact that once the sample $\mathbf{X}_1,\ldots,\mathbf{X}_N$ is generated, the functions $\widehat{\ell}_j(\mathbf{u})$, $j = 0,\ldots,M$ become known explicitly, provided that the sample functions $\{H_j(\mathbf{X};\mathbf{u}_2)\}$ are explicitly available for any $\mathbf{u}_2$. Recall that if $H_j(\mathbf{X};\mathbf{u}_2)$ is available only for some fixed in advance $\mathbf{u}_2$, rather than simultaneously for all values $\mathbf{u}_2$, one can apply stochastic approximation algorithms instead of the stochastic counterpart method. Note that in the case where the $\{H_j(\cdot)\}$ do not depend on $\mathbf{u}_2$, one can solve the program $(\widehat{\mathrm{P}}_N)$ (from a single simulation run) using the SF method, provided that the trust region of the program $(\widehat{\mathrm{P}}_N)$ does not exceed the one defined in (7.27). If this is not the case, one needs to use iterative gradient-type methods, which do not involve likelihood ratios.

The algorithm for estimating the optimal solution, $\mathbf{u}^*$, of the program (P_0) via the stochastic counterpart $(\widehat{\mathrm{P}}_N)$ can be written as follows:

Algorithm 7.3.1 (Estimation of $\mathbf{u}^*$)

1. *Generate a random sample* $\mathbf{X}_1,\ldots,\mathbf{X}_N$ *from* $f(\mathbf{x};\mathbf{v}_1)$.
2. *Calculate the functions* $H_j(\mathbf{X}_i;\mathbf{u}_2)$, $j = 0,\ldots,M$, $i = 1,\ldots,N$ *via simulation.*
3. *Solve the program* $(\widehat{\mathrm{P}}_N)$ *by standard mathematical programming methods.*
4. *Return the resultant optimal solution,* $\widehat{\mathbf{u}^*}$ *of* $(\widehat{\mathrm{P}}_N)$, *as an estimate of* $\mathbf{u}^*$.

The third step of Algorithm 7.3.1 typically calls for iterative numerical procedures, which may require, in turn, calculation of the functions $\widehat{\ell}_j(\mathbf{u})$, $j = 0, \ldots, M$, and their gradients (and possibly Hessians), for multiple values of the parameter vector $\mathbf{u}$. Our extensive simulation studies for typical DESS with sizes up to 100 decision variables show that the optimal solution $\widehat{\mathbf{u}^*}$ of the program $(\widehat{\mathrm{P}}_N)$ constitutes a reliable estimator of the true optimal solution, $\mathbf{u}^*$, provided that the program $(\widehat{\mathrm{P}}_N)$ is convex (see [18] and the Appendix), the trust region is not too large, and the sample size N is quite large (on the order of 1000 or more).

7.4 SENSITIVITY ANALYSIS OF DEDS

Let $X_1, X_2, \ldots$ be an input sequence of m-dimensional random vectors driving an output process $\{H_t, t = 0, 1, 2, \ldots\}$. That is, $H_t = H_t(\mathbf{X}_t)$ for some function H_t, where the vector $\mathbf{X}_t = (X_1, X_2, \ldots, X_t)$ represents the history of the input process up to time t. Let the pdf of $\mathbf{X}_t$ be given by $f_t(\mathbf{x}_t; \mathbf{u})$, which depends on some parameter vector $\mathbf{u}$. Assume that $\{H_t\}$ is a regenerative process with a regenerative cycle of length τ. Typical examples are an ergodic Markov chain and the waiting time process in the $GI/G/1$ system. In both cases (see Section 4.3.2.2) the expected steady-state performance, $\ell(\mathbf{u})$, can be written as

$$\ell(\mathbf{u}) = \frac{\mathbb{E}_{\mathbf{u}}[R]}{\mathbb{E}_{\mathbf{u}}[\tau]} = \frac{\mathbb{E}_{\mathbf{u}}\left[\sum_{t=1}^{\tau} H_t(\mathbf{X}_t)\right]}{\mathbb{E}_{\mathbf{u}}[\tau]}\,, \tag{7.66}$$

where R is the reward during a cycle. As for static models, we show here how to estimate from a *single simulation run* the performance $\ell(\mathbf{u})$, and the derivatives $\nabla^k \ell(\mathbf{u})$, $k = 1, 2 \ldots$, for different values of $\mathbf{u}$.

Consider first the estimation of $\ell_R(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[R]$ when the $\{X_i\}$ are iid with pdf $f(x; \mathbf{u})$; thus, $f_t(\mathbf{x}_t) = \prod_{i=1}^{t} f(x_i)$. Let $g(x)$ be any importance sampling pdf, and let $g_t(\mathbf{x}_t) = \prod_{i=1}^{t} g(x_i)$. It will be shown that $\ell_R(\mathbf{u})$ can be represented as

$$\ell_R(\mathbf{u}) = \mathbb{E}_g\left[\sum_{t=1}^{\tau} H_t(\mathbf{X}_t)\, W_t(\mathbf{X}_t; \mathbf{u})\right], \tag{7.67}$$

where $\mathbf{X}_t \sim g_t(\mathbf{x}_t)$ and $W_t(\mathbf{X}_t; \mathbf{u}) = f_t(\mathbf{x}_t; \mathbf{u})/g_t(\mathbf{x}_t) = \prod_{j=1}^{t} f(X_j; \mathbf{u})/g(X_j)$. To proceed, we write

$$\sum_{t=1}^{\tau} H_t = \sum_{t=1}^{\infty} H_t\, I_{\{\tau \geqslant t\}}\,. \tag{7.68}$$

Since $\tau = \tau(\mathbf{X}_t)$ is completely determined by $\mathbf{X}_t$, the indicator $I_{\{\tau \geqslant t\}}$ can be viewed as a function of $\mathbf{x}_t$; we write $I_{\{\tau \geqslant t\}}(\mathbf{x}_t)$. Accordingly, the expectation of $H_t\, I_{\{\tau \geqslant t\}}$ is

$$\begin{aligned}
\mathbb{E}_{\mathbf{u}}[H_t\, I_{\{\tau \geqslant t\}}] &= \int H_t(\mathbf{x}_t)\, I_{\{\tau \geqslant t\}}(\mathbf{x}_t)\, f_t(\mathbf{x}_t; \mathbf{u})\, \mathrm{d}\mathbf{x}_t \\
&= \int H_t(\mathbf{x}_t)\, I_{\{\tau \geqslant t\}}(\mathbf{x}_t)\, W_t(\mathbf{x}_t; \mathbf{u})\, g_t(\mathbf{x}_t)\, \mathrm{d}\mathbf{x}_t \\
&= \mathbb{E}_g[H_t(\mathbf{X}_t)\, I_{\{\tau \geqslant t\}}(\mathbf{X}_t)\, W_t(\mathbf{X}_t; \mathbf{u})]\,.
\end{aligned} \tag{7.69}$$

The result (7.67) follows by combining (7.68) and (7.69). For the special case where $H_t \equiv 1$, (7.67) reduces to

$$\mathbb{E}_{\mathbf{u}}[\tau] = \mathbb{E}_g\left[\sum_{t=1}^{\tau} W_t\right],$$

abbreviating $W_t(\mathbf{X}_t; \mathbf{u})$ to W_t. Derivatives of (7.67) can be presented in a similar form. In particular, under standard regularity conditions ensuring the interchangeability of the differentiation and the expectation operators, one can write

$$\nabla^k \ell_R(\mathbf{u}) = \mathbb{E}_g \left[\sum_{t=1}^{\tau} H_t \nabla^k W_t \right] = \mathbb{E}_g \left[\sum_{t=1}^{\tau} H_t \, \mathcal{S}_t^{(k)} \, W_t \right] , \tag{7.70}$$

where $\mathcal{S}_t^{(k)}$ is the k-th order score function corresponding to $f_t(\mathbf{x}_t; \mathbf{u})$, as in (7.7).

Now let $\{X_{11}, \ldots, X_{\tau_1 1}, \ldots, X_{1N}, \ldots, X_{\tau_N N}\}$ be a sample of N regenerative cycles from the pdf $g(x)$. Then, using (7.70), we can estimate $\nabla^k \ell_R(\mathbf{u})$, $k = 0, 1, \ldots$ from a *single simulation run* as

$$\widehat{\nabla^k \ell_R}(\mathbf{u}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{\tau_i} H_{ti} \, \mathcal{S}_{ti}^{(k)} \, W_{ti} , \tag{7.71}$$

where $W_{ti} = \prod_{j=1}^{t} \frac{f(X_{ji};\mathbf{u})}{g(X_{ji})}$ and $X_{ji} \sim g(x)$. Notice that here $\widehat{\nabla^k \ell_R}(\mathbf{u}) = \nabla^k \widehat{\ell_R}(\mathbf{u})$. For the special case where $g(\mathbf{x}) = f(\mathbf{x}; \mathbf{u})$, that is, when using the original pdf $f(\mathbf{x}; \mathbf{u})$, one has

$$\nabla^k \ell_R(\mathbf{u}) = \mathbb{E}_{\mathbf{u}} \left[\sum_{t=1}^{\tau} H_t \, \mathcal{S}_t^{(k)} \right] . \tag{7.72}$$

For $k = 1$, writing $\mathcal{S}_t$ for $\mathcal{S}_t^{(1)}$, the score function process $\{\mathcal{S}_t\}$ is given by

$$\mathcal{S}_t = \sum_{j=1}^{t} \nabla \ln f(X_j; \mathbf{u}) . \tag{7.73}$$

■ **EXAMPLE 7.10**

Let $X \sim \mathsf{G}(p)$. That is, $f(x; p) = p(1-p)^{x-1}$, $x = 1, 2, \ldots$. Then (see also Table 7.1)

$$\mathcal{S}_t = \frac{\partial}{\partial p} \ln f_t(\mathbf{X}_t; p) = \frac{t - p \sum_{j=1}^{t} X_j}{p(1-p)} .$$

■ **EXAMPLE 7.11**

Let $X \sim \mathsf{Gamma}(\alpha, \lambda)$. That is, $f(x; \lambda, \alpha) = \frac{\lambda^\alpha x^{\alpha-1} e^{-\lambda x}}{\Gamma(\alpha)}$ for $x > 0$. Suppose we are interested in the sensitivities with respect to λ. Then

$$\mathcal{S}_t = \frac{\partial}{\partial \lambda} \ln f_t(\mathbf{X}_t; \lambda, \alpha) = t \, \alpha \, \lambda^{-1} - \sum_{i=1}^{t} X_i .$$

Let us return now to the estimation of $\ell(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[R]/\mathbb{E}_{\mathbf{u}}[\tau]$ and its sensitivities. In view of (7.70) and the fact that $\tau = \sum_{t=1}^{\tau} 1$ can be viewed as a special case of (7.67), with $H_t = 1$, one can write $\ell(\mathbf{u})$ as

$$\ell(\mathbf{u}) = \frac{\mathbb{E}_g[\sum_{t=1}^{\tau} H_t \, W_t]}{\mathbb{E}_g[\sum_{t=1}^{\tau} W_t]} \tag{7.74}$$

and by direct differentiation of (7.74) write $\nabla \ell(\mathbf{u})$ as

$$\nabla \ell(\mathbf{u}) = \frac{\mathbb{E}_g[\sum_{t=1}^{\tau} H_t \nabla W_t]}{\mathbb{E}_g[\sum_{t=1}^{\tau} W_t]} - \frac{\mathbb{E}_g[\sum_{t=1}^{\tau} H_t W_t]}{\mathbb{E}_g[\sum_{t=1}^{\tau} W_t]} \cdot \frac{\mathbb{E}_g[\sum_{t=1}^{\tau} \nabla W_t]}{\mathbb{E}_g[\sum_{t=1}^{\tau} W_t]} \tag{7.75}$$

(observe that $W_t = W_t(\mathbf{X}_t, \mathbf{u})$ is a function of $\mathbf{u}$ but $H_t = H_t(\mathbf{X}_t)$ is not). Observe also that above, $\nabla W_t = W_t \mathcal{S}_t$. Higher-order partial derivatives with respect to parameters of interest can then be obtained from (7.75). Utilizing (7.74) and (7.75), one can estimate $\ell(\mathbf{u})$ and $\nabla \ell(\mathbf{u})$, for all $\mathbf{u}$, as

$$\widehat{\ell}(\mathbf{u}) = \frac{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} H_{ti} W_{ti}}{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti}} \tag{7.76}$$

and

$$\widehat{\nabla \ell}(\mathbf{u}) = \frac{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} H_{ti} W_{ti} \mathcal{S}_{ti}}{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti}} - \frac{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} H_{ti} W_{ti}}{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti}} \cdot \frac{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti} \mathcal{S}_{ti}}{\sum_{i=1}^{N} \sum_{t=1}^{\tau_i} W_{ti}}, \tag{7.77}$$

respectively, and similarly for higher-order derivatives. Notice again that in this case, $\widehat{\nabla \ell}(\mathbf{u}) = \nabla \widehat{\ell}(\mathbf{u})$. The algorithm for estimating the gradient $\nabla \ell(\mathbf{u})$ at different values of $\mathbf{u}$ using a single simulation run can be written as follows.

Algorithm 7.4.1 ($\nabla \ell(\mathbf{u})$ Estimation)

1. *Generate a random sample* $\{X_1, \ldots, X_T\}$, $T = \sum_{i=1}^{N} \tau_i$, *from* $g(x)$.
2. *Generate the output processes* $\{H_t\}$ *and* $\{\nabla W_t\} = \{W_t \mathcal{S}_t\}$.
3. *Calculate* $\widehat{\nabla \ell}(\mathbf{u})$ *from (7.77).*

Confidence intervals (regions) for the sensitivities $\nabla^k \ell(\mathbf{u})$, $k = 0, 1$, utilizing the SF estimators $\nabla^k \widehat{\ell}(\mathbf{u})$, $k = 0, 1$, can be derived analogously to those for the standard regenerative estimator of Chapter 4 and are left as an exercise.

■ **EXAMPLE 7.12 Waiting Time**

The waiting time process in a $GI/G/1$ queue is driven by sequences of interarrival times $\{A_t\}$ and service times $\{S_t\}$ via the Lindley equation

$$H_t = \max\{H_{t-1} + S_t - A_t, 0\}, \quad t = 1, 2, \ldots \tag{7.78}$$

with $H_0 = 0$; see (4.30) and Problem 5.3. Writing $X_t = (S_t, A_t)$, the $\{X_t, t = 1, 2, \ldots\}$ are iid. The process $\{H_t, t = 0, 1, \ldots\}$ is a regenerative process, which regenerates every time $H_t = 0$. Let $\tau > 0$ denote the first such time, and let H denote the steady-state waiting time. We wish to estimate the steady-state performance

$$\ell = \mathbb{E}[H] = \frac{\mathbb{E}[\sum_{t=1}^{\tau} H_t]}{\mathbb{E}[\tau]}.$$

Consider, for instance, the case where $S \sim \text{Exp}(\mu)$, $A \sim \text{Exp}(\lambda)$, and S and A are independent. Thus, H is the steady-state waiting time in the $M/M/1$ queue, and $\mathbb{E}[H] = \lambda/(\mu(\mu - \lambda))$ for $\mu > \lambda$; see, for example, [5]. Suppose we carry out the simulation using the service rate $\tilde{\mu}$ and wish to estimate $\ell(\mu) = \mathbb{E}[H]$ for different

values of μ using the same simulation run. Let $(S_1, A_1), \ldots, (S_\tau, A_\tau)$ denote the service and interarrival times in the first cycle, respectively. Then, for the first cycle

$$W_t = W_{t-1} \frac{\mu \mathrm{e}^{-\mu S_t}}{\tilde{\mu} \mathrm{e}^{-\tilde{\mu} S_t}}, \quad t = 1, 2, \ldots, \tau \quad (W_0 = 1),$$

$$\mathcal{S}_t = \mathcal{S}_{t-1} + \frac{1}{\mu} - S_t, \quad t = 1, 2, \ldots, \tau \quad (\mathcal{S}_0 = 0),$$

and H_t is as given in (7.78). From these, the sums $\sum_{t=1}^{\tau} H_t W_t$, $\sum_{t=1}^{\tau} W_t$, $\sum_{t=1}^{\tau} W_t \mathcal{S}_t$, and $\sum_{t=1}^{\tau} H_t W_t \mathcal{S}_t$ can be computed. Repeating this for the subsequent cycles, one can estimate $\ell(\mu)$ and $\nabla \ell(\mu)$ from (7.76) and (7.77), respectively. Figure 7.4 displays the estimates and true values for $1.5 \leqslant \mu \leqslant 5.5$ using a single simulation run of $N = 10^5$ cycles. The simulation was carried out under the service rate $\tilde{\mu} = 2$ and arrival rate $\lambda = 1$. We see that both $\ell(\mu)$ and $\nabla \ell(\mu)$ are estimated accurately over the whole range. Note that for $\mu < 2$ the confidence interval for $\ell(\mu)$ grows rapidly wider. The estimation should not be extended much below $\mu = 1.5$, as the importance sampling will break down, resulting in unreliable estimates.

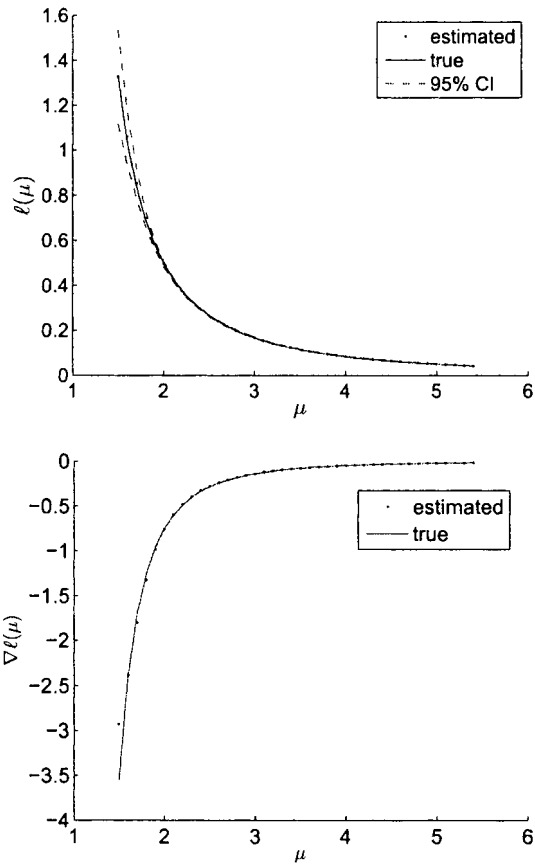


Figure 7.4 Estimated and true values for the expected steady-state waiting time and its derivative as a function of μ.

Although (7.76) and (7.77) were derived for the case where the $\{X_i\}$ are iid, much of the theory can be readily modified to deal with the dependent case. As an example, consider the case where $X_1, X_2, \ldots$ form an ergodic Markov chain and R is of the form

$$R = \sum_{t=1}^{\tau} c_{X_{t-1},X_t} , \tag{7.79}$$

where c_{ij} is the cost of going from state i to j and R represents the cost accrued in a cycle of length τ. Let $\mathbf{P} = (p_{ij})$ be the one-step transition matrix of the Markov chain. Following reasoning similar to that for (7.67) and defining $H_t = c_{X_{t-1},X_t}$, we see that

$$\mathbb{E}_{\mathbf{P}}[R] = \mathbb{E}_{\widetilde{\mathbf{P}}}\left[\sum_{t=1}^{\tau} H_t W_t\right],$$

where $\widetilde{\mathbf{P}} = (\widetilde{p}_{ij})$ is another transition matrix, and

$$W_t = W_t(\mathbf{X}_t; \mathbf{P}, \widetilde{\mathbf{P}}) = \prod_{k=1}^{t} \frac{p_{X_{k-1},X_k}}{\widetilde{p}_{X_{k-1},X_k}}$$

is the likelihood ratio. The pdf of $\mathbf{X}_t$ is given by

$$f_t(\mathbf{x}_t; \mathbf{P}) = \prod_{k=1}^{t} p_{X_{k-1},X_k} .$$

The score function can again be obtained by taking the derivative of the logarithm of the pdf. Since, $\mathbb{E}_{\mathbf{P}}[\tau] = \mathbb{E}_{\widetilde{\mathbf{P}}}[\sum_{t=1}^{\tau} W_t]$, the long-run average cost $\ell(\mathbf{P}) = \mathbb{E}_{\mathbf{P}}[R]/\mathbb{E}_{\mathbf{P}}[\tau]$ can be estimated via (7.76) — and its derivatives by (7.77) — simultaneously for various $\mathbf{P}$ using a single simulation run under $\widetilde{\mathbf{P}}$.

■ **EXAMPLE 7.13 Markov Chain: Example 4.8 (Continued)**

Consider again the two-state Markov chain with transition matrix $\mathbf{P} = (p_{ij})$ and cost matrix C given by

$$\mathbf{P} = \begin{pmatrix} p_1 & 1-p_1 \\ p_2 & 1-p_2 \end{pmatrix} = \begin{pmatrix} \mathbf{p} & \mathbf{1}-\mathbf{p} \end{pmatrix}$$

and

$$C = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix},$$

respectively, where $\mathbf{p}$ denotes the vector $(p_1, p_2)^T$. Our goal is to estimate $\ell(\mathbf{p})$ and $\nabla\ell(\mathbf{p})$ using (7.76) and (7.77) for various $\mathbf{p}$ from a single simulation run under $\widetilde{\mathbf{p}} = (\frac{1}{2}, \frac{1}{5})^T$. Assume, as in Example 4.8, that starting from state 1, we obtain the sample trajectory $(x_0, x_1, x_2, \ldots, x_{10}) = (1, 2, 2, 2, 1, 2, 1, 1, 2, 2, 1)$, which has four cycles with lengths $\tau_1 = 4$, $\tau_2 = 2$, $\tau_3 = 1$, $\tau_4 = 3$ and corresponding transition probabilities $(p_{12}, p_{22}, p_{22}, p_{21})$; (p_{12}, p_{21}); (p_{11}); (p_{12}, p_{22}, p_{21}). The cost in the first cycle is given by (7.79). We consider the cases (1) $\mathbf{p} = \widetilde{\mathbf{p}} = (\frac{1}{2}, \frac{1}{5})^T$ and (2) $\mathbf{p} = (\frac{1}{5}, \frac{1}{2})^T$. The transition matrices for the two cases are

$$\widetilde{\mathbf{P}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{5} & \frac{4}{5} \end{pmatrix} \quad \text{and} \quad \mathbf{P} = \begin{pmatrix} \frac{1}{5} & \frac{4}{5} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} .$$

Note that the first case pertains to the nominal Markov chain.

In the first cycle, costs $H_{11} = 1, H_{21} = 3, H_{31} = 3$, and $H_{41} = 2$ are incurred. The likelihood ratios under case (2) are $W_{11} = \frac{p_{12}}{\tilde{p}_{12}} = 8/5$, $W_{21} = W_{11}\frac{p_{22}}{\tilde{p}_{22}} = 1$, $W_{31} = W_{21}\frac{p_{22}}{\tilde{p}_{22}} = \frac{5}{8}$ and $W_{41} = W_{31}\frac{p_{21}}{\tilde{p}_{21}} = \frac{25}{16}$, while in case (1) they are all 1. Next, we derive the score functions (in the first cycle) with respect to p_1 and p_2. Note that

$$f_4(\mathbf{x}_4; \mathbf{p}) = p_{12}\, p_{22}^2\, p_{21} = (1 - p_1)(1 - p_2)^2\, p_2 \,.$$

It follows that in case (2)

$$\frac{\partial}{\partial p_1} \ln f_4(\mathbf{x}_4; \mathbf{p}) = \frac{-1}{1 - p_1} = -\frac{5}{4}$$

and

$$\frac{\partial}{\partial p_2} \ln f_4(\mathbf{x}_4; \mathbf{p}) = \frac{-2}{1 - p_2} + \frac{1}{p_2} = -2 \,,$$

so that the score function at time $t = 4$ in the first cycle is given by $\mathcal{S}_{41} = (-\frac{5}{4}, -2)$. Similarly, $\mathcal{S}_{31} = (-\frac{5}{4}, -4)$, $\mathcal{S}_{21} = (-\frac{5}{4}, -2)$, and $\mathcal{S}_{11} = (-\frac{5}{4}, 0)$. The quantities for the other cycles are derived in the same way, and the results are summarized in Table 7.3.

Table 7.3 Summary of costs, likelihood ratios, and score functions.

i	H_{ti}	W_{ti} (case 2)	$\mathcal{S}_{ti}$ (case 1)	$\mathcal{S}_{ti}$ (case 2)
1	1, 3, 3, 2	$\frac{8}{5}, 1, \frac{5}{8}, \frac{25}{16}$	$(-2,0), (-2,-\frac{5}{4}), (-2,-\frac{5}{2}), (-2,\frac{5}{2})$	$(-\frac{5}{4},0), (-\frac{5}{4},-2), (-\frac{5}{4},-4), (-\frac{5}{4},-2)$
2	1, 2	$\frac{8}{5}, 4$	$(-2,0), (-2,5)$	$(-\frac{5}{4},0), (-\frac{5}{4},2)$
3	0	$\frac{2}{5}$	$(2,0)$	$(5,0)$
4	1, 3, 2	$\frac{8}{5}, 1, \frac{5}{2}$	$(-2,0), (-2,-\frac{5}{4}), (-2,\frac{15}{4})$	$(-\frac{5}{4},0), (-\frac{5}{4},-2), (-\frac{5}{4},0)$

By substituting these values in (7.76) and (7.77), the reader can verify that $\widehat{\ell}(\tilde{\mathbf{p}}) = 1.8$, $\widehat{\ell}(\mathbf{p}) \approx 1.81$, $\widehat{\nabla\ell}(\tilde{\mathbf{p}}) = (-0.52, -0.875)$, and $\widehat{\nabla\ell}(\mathbf{p}) \approx (0.22, -1.23)$.

PROBLEMS

7.1 Consider the unconstrained minimization program

$$\min_u \ell(u) = \min_u \left\{ \mathbb{E}_u[X] + \frac{b}{u} \right\}, \quad u \in (0,1) \,, \tag{7.80}$$

where $X \sim \mathsf{Ber}(u)$.

a) Show that the stochastic counterpart of $\nabla\ell(u) = 0$ can be written (see (7.18)) as

$$\widehat{\nabla\ell}(u) = \nabla\widehat{\ell}(u; v) = \frac{1}{v}\frac{1}{N}\sum_{i=1}^{N} X_i - \frac{b}{u^2} = 0 \,, \tag{7.81}$$

where $X_1, \ldots, X_N$ is a random sample from $\mathsf{Ber}(v)$.

b) Assume that the sample $\{0,1,0,0,1,0,0,1,1,1,0,1,0,1,1,0,1,0,1,1\}$ was generated from $\mathsf{Ber}(v = 1/2)$. Show that the optimal solution u^* is estimated as

$$\widehat{u^*} = \left(\frac{b}{1.1}\right)^{1/2} .$$

7.2 Consider the unconstrained minimization program

$$\min_u \ell(u) = \min_u \{\mathbb{E}_u[X] + bu\},\ u \in (0.5, 2.0)\ , \tag{7.82}$$

where $X \sim \mathsf{Exp}(u)$. Show that the stochastic counterpart of $\nabla\ell(u) = -\frac{1}{u^2} + b = 0$ can be written (see (7.20)) as

$$\nabla\widehat{\ell}(u;v) = \frac{1}{N}\sum_{i=1}^{N} X_i \frac{\mathrm{e}^{-uX_i}(1 - u\,X_i)}{v\,\mathrm{e}^{-vX_i}} + b = 0\ , \tag{7.83}$$

where $X_1, \ldots, X_n$ is a random sample from $\mathsf{Exp}(v)$.

7.3 Prove (7.25).

7.4 Show that $\nabla^k W(\mathbf{x};\mathbf{u},\mathbf{v}) = \mathbb{S}^{(k)}(\mathbf{u};\mathbf{x})\, W(\mathbf{x};\mathbf{u},\mathbf{v})$ and hence prove (7.16).

7.5 Let $X_i \sim \mathsf{N}(u_i, \sigma_i^2)$, $i = 1, \ldots, n$ be independent random variables. Suppose we are interested in sensitivities with respect to $\mathbf{u} = (u_1, \ldots, u_n)$ only. Show that, for $i = 1, \ldots, n$,

$$\mathbb{E}_{\mathbf{v}}[W^2] = \exp\left(\sum_{i=1}^{n} \frac{(u_i - v_i)^2}{\sigma_i^2}\right)$$

and

$$[\mathbb{S}^{(1)}(\mathbf{u};\mathbf{X})]_i = \sigma_i^{-2}(x_i - u_i)\ .$$

7.6 Let the components X_i, $i = 1, \ldots, n$ of a random vector $\mathbf{X}$ be independent, and distributed according to the exponential family

$$f_i(x_i;\mathbf{u}_i) = c_i(\mathbf{u}_i)\,\mathrm{e}^{b_i(\mathbf{u}_i)t_i(x_i)}\,h_i(x_i)\ ,$$

where $b_i(\mathbf{u}_i)$, $t_i(x_i)$ and $h_i(x_i)$ are real-valued functions and $c_i(\mathbf{u}_i)$ is normalization constant. The corresponding pdf of $\mathbf{X}$ is given by

$$f(\mathbf{x};\mathbf{u}) = c(\mathbf{u}) \exp\left(\sum_{i=1}^{n} b_i(\mathbf{u}_i)t_i(x_i)\right) h(\mathbf{x})\ ,$$

where $\mathbf{u} = (\mathbf{u}_1^T, \ldots, \mathbf{u}_n^T)$, $c(\mathbf{u}) = \prod_{i=1}^n c_i(\mathbf{u}_i)$, and $h(\mathbf{x}) = \prod_{i=1}^n h_i(x_i)$.

a) Show that $\mathrm{Var}_{\mathbf{v}}(HW) = \frac{c(\mathbf{u})^2}{c(\mathbf{v})\,c(\mathbf{w})}\,\mathbb{E}_{\mathbf{w}}[H^2] - \ell(\mathbf{u})^2$, where $\mathbf{w}$ is determined by $b_i(\mathbf{w}_i) = 2b_i(\mathbf{u}_i) - b_i(\mathbf{v}_i)$, $i = 1, \ldots, n$.

b) Show that

$$\mathbb{E}_{\mathbf{v}}[H^2W^2] = \mathbb{E}_{\mathbf{v}}[W^2]\,\mathbb{E}_{\mathbf{w}}[H^2]\ .$$

7.7 Consider the exponential pdf $f(x;u) = u\exp(-ux)$. Show that if $H(x)$ is a monotonically increasing function, then the expected performance $\ell(u) = \mathbb{E}_u[H(X)]$ is a monotonically decreasing convex function of $u \in (0,\infty)$.

7.8 Let $X \sim \mathsf{N}(u, \sigma^2)$. Suppose that σ is known and fixed. For a given u, consider the function

$$\mathcal{L}(v) = \mathbb{E}_v[H^2 W^2] \,.$$

a) Show that if $\mathbb{E}_u[H^2] < \infty$ for all $u \in \mathbb{R}$, then $\mathcal{L}(v)$ is convex and continuous on $\mathbb{R}$. Show further that if, additionally, $\mathbb{E}_{u_n}[H^2] > 0$ for any u, then $\mathcal{L}(v)$ has a unique minimizer, v^*, over $\mathbb{R}$.

b) Show that if $H^2(x)$ is monotonically increasing on $\mathbb{R}$, then $v^* > u$.

7.9 Let $X \sim \mathsf{N}(u, \sigma^2)$. Suppose that u is known, and consider the parameter σ. Note that the resulting exponential family is not of canonical form (A.9). However, parameterizing it by $\theta = \sigma^{-2}$ transforms it into canonical form, with $t(x) = -(x-u)^2/2$ and $c(\theta) = (2\pi)^{-1/2}\theta^{1/2}$.

a) Show that

$$\mathbb{E}_\eta[W^2] = \frac{\theta}{\eta^{1/2}(2\theta - \eta)^{1/2}} \,,$$

provided that $0 < \eta < 2\theta$.

b) Show that, for a given θ, the function

$$\mathcal{L}(\eta) = \mathbb{E}_\eta[H^2 W^2]$$

has a unique minimizer, η^*, on the interval $(0, 2\theta)$, provided that the expectation, $\mathbb{E}_\eta[H^2]$, is finite for all $\eta \in (0, 2\theta)$ and does not tend to 0 as η approaches 0 or 2θ. (Notice that this implies that the corresponding optimal value, $\sigma^* = \eta^{*-1/2}$, of the reference parameter, σ, is also unique.)

c) Show that if $H^2(x)$ is strictly convex on $\mathbb{R}$, then $\eta^* < \theta$. (Notice that this implies that $\sigma^* > \sigma$.)

7.10 Consider the performance

$$H(X_1, X_2; u_3, u_4) = \min\{\max(X_1, u_3), \max(X_2, u_4)\} \,,$$

where X_1 and X_2 have continuous densities $f(x_1; u_1)$ and $f(x_2; u_2)$, respectively. If we let $Y_1 = \max(X_1, u_3)$ and $Y_2 = \max(X_2, u_4)$ and write the performance as $\min(Y_1, Y_2)$, then Y_1 and Y_2 would take values u_3 and u_4 with nonzero probability. Hence the random vector $\mathbf{Y} = (Y_1, Y_2)$ would not have a density function at point (u_3, u_4), since its distribution is a mixture of continuous and discrete ones. Consequently, the push-out method would fail in its current form. To overcome this difficulty, we carry out a transformation. We first write Y_1 and Y_2 as

$$Y_1 = u_3 \max\left(\frac{X_1}{u_3}, 1\right), \quad Y_2 = u_4 \max\left(\frac{X_2}{u_4}, 1\right)$$

and then replace $\mathbf{X} = (X_1, X_2)$ by the random vector $\widetilde{\mathbf{X}} = (\widetilde{X}_1, \widetilde{X}_2)$, where

$$\widetilde{X}_1 = \max\left(\frac{X_1}{u_3}, 1\right)$$

and

$$\widetilde{X}_2 = \max\left(\frac{X_2}{u_4}, 1\right).$$

Prove that the density of the random vector $(\widetilde{X}_1, \widetilde{X}_2)$ is differentiable with respect to the variables (u_3, u_4), provided that both $\widetilde{X}_1$ and $\widetilde{X}_2$ are greater than 1.

7.11 Delta method. Let $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_m)$ be random (column) vectors, with $\mathbf{Y} = \mathbf{g}(\mathbf{X})$ for some mapping $\mathbf{g}$ from $\mathbb{R}^n$ to $\mathbb{R}^m$. Let $\Sigma_{\mathbf{X}}$ and $\Sigma_{\mathbf{Y}}$ denote the corresponding covariance matrices. Suppose that $\mathbf{X}$ is close to its mean μ. A first-order Taylor expansion of $\mathbf{g}$ around μ gives

$$\mathbf{Y} \approx \mathbf{g}(\mu) + J_{\mu}(\mathbf{g})(\mathbf{X} - \mu),$$

where $J_{\mu}(\mathbf{g})$ is the matrix of Jacobi of $\mathbf{g}$ (the matrix whose (i, j)-th entry is the partial derivative $\partial g_i / \partial x_j$) evaluated at μ. Show that, as a consequence,

$$\Sigma_{\mathbf{Y}} \approx J_{\mu}(\mathbf{g}) \Sigma_{\mathbf{X}} J_{\mu}(\mathbf{g})^T .$$

This is called the *delta method* in statistics.

Further Reading

The SF method in the simulation context appears to have been discovered and rediscovered independently, starting in the late 1960s. The earlier work on SF appeared in Aleksandrov, Sysoyev, and Shemeneva [1] in 1968 and Rubinstein [14] in 1969. Motivated by the pioneering paper of Ho, Eyler, and Chien [6] on *infinitesimal perturbation analysis* (IPA) in 1979, the SF method was rediscovered at the end of the 1980s by Glynn [4] in 1990 and independently in 1989 by Reiman and Weiss [12], who called it the *likelihood ratio method.* Since then, both the IPA and SF methods have evolved over the past decade or so and have now reached maturity; see Glasserman [3], Pflug [11], Rubinstein and Shapiro [18], and Spall [20].

To the best of our knowledge, the stochastic counterpart method in the simulation context was first suggested by Rubinstein in his PhD thesis [14]. It was applied there to estimate the optimal parameters in a complex simulation-based optimization model. It was shown numerically that the *off-line* stochastic counterpart method produces better estimates than the standard *on-line* stochastic approximation. For some later work on the stochastic counterpart method and stochastic approximation, see [15]. Alexander Shapiro should be credited for developing theoretical foundations for stochastic programs and, in particular, for the stochastic counterpart method. For relevant references, see Shapiro's elegant paper [19] and also [17, 18]. As mentioned, Geyer and Thompson [2] independently discovered the stochastic counterpart method in the early 1990s, and used it to make statistical inference in a particular unconstrained setting.

REFERENCES

1. V. M Aleksandrov, V. I. Sysoyev, and V. V. Shemeneva. Stochastic optimization. *Engineering Cybernetics*, 5:11–16, 1968.
2. C. J. Geyer and E. A. Thompson. Annealing Markov chain Monte-Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90:909–920, 1995.
3. P. Glasserman. *Gradient Estimation via Perturbation Analysis*. Kluwer, Norwell, Mass., 1991.
4. P. W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.
5. D. Gross and C. M. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 2nd edition, 1985.

6. Y. C. Ho, M. A. Eyler, and T. T. Chien. A gradient technique for general buffer storage design in a serial production line. *International Journal on Production Research*, 17(6):557–580, 1979.

7. J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.

8. A. J. Kleywegt and A. Shapiro. Stochastic optimization. In G. Salvendy, editor, *Handbook of Industrial Engineering*, pages 2625–2650, New York, 2001. John Wiley & Sons.

9. A. J. Kleywegt, A. Shapiro, and T. Homem de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12:479–502, 2001.

10. H. J. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag, New York, 1978.

11. G. Ch. Pflug. *Optimization of Stochastic Models*. Kluwer, Boston, 1996.

12. M. I. Reiman and A. Weiss. Sensitivity analysis for simulations via likelihood ratios. *Operations Research*, 37(5):830–844, 1989.

13. H. Robbins and S. Monro. Stochastic approximation methods. *Annals of Mathematical Statistics*, 22:400–407, 1951.

14. R. Y. Rubinstein. *Some Problems in Monte Carlo Optimization*. PhD thesis, University of Riga, Latvia, 1969. (In Russian).

15. R. Y. Rubinstein. *Monte Carlo Optimization Simulation and Sensitivity of Queueing Network*. John Wiley & Sons, New York, 1986.

16. R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling*. John Wiley & Sons, New York, 1998.

17. R. Y. Rubinstein and A. Shapiro. Optimization of static simulation models by the score function method. *Mathematics and Computers in Simulation*, 32:373–392, 1990.

18. R. Y. Rubinstein and A. Shapiro. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization via the Score Function Method*. John Wiley & Sons, New York, 1993.

19. A. Shapiro. Simulation based optimization: convergence analysis and statistical inference. *Stochastic Models*, 12:425–454, 1996.

20. J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons, New York, 2003.

CHAPTER 8

THE CROSS-ENTROPY METHOD

8.1 INTRODUCTION

The *cross-entropy* (CE) method [31] is a relatively new Monte Carlo technique for both estimation and optimization. In the estimation setting, the CE method provides an adaptive way to find the optimal importance sampling distribution for quite general problems. By formulating an optimization problem as an estimation problem, the CE method becomes a general and powerful stochastic search algorithm. The method is based on a simple iterative procedure where each iteration contains two phases: (a) generate a random data sample (trajectories, vectors, etc.) according to a specified mechanism; (b) update the parameters of the random mechanism on the basis of the data in order to produce a better sample in the next iteration.

The CE method has its origins in an adaptive algorithm for rare-event estimation based on *variance minimization* (VM) [26]. This procedure was soon modified [27] to an adaptive algorithm for both rare-event estimation and combinatorial optimization, where the original VM program was replaced by a similar CE minimization program. In this chapter we present a general introduction to the CE method. For a comprehensive treatment we refer to [31].

The rest of this chapter is organized as follows. Section 8.2 presents a general CE algorithm for the estimation of rare-event probabilities, while Section 8.3 introduces a slight modification of this algorithm for solving combinatorial optimization problems. We discuss applications of the CE method to several such problems, such as the max-cut problem and the TSP, and provide supportive numerical results on the performance of the algorithm.

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

Finally, in Sections 8.7 and 8.8 we show how the CE method can deal with continuous and noisy optimization problems, respectively.

8.2 ESTIMATION OF RARE-EVENT PROBABILITIES

In this section we apply the CE method in the context of efficient estimation of small probabilities. Consider, in particular, the estimation of

$$\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}}\left[I_{\{S(\mathbf{X}) \geqslant \gamma\}}\right] \tag{8.1}$$

for some fixed level γ. Here $S(\mathbf{X})$ is the sample performance, $\mathbf{X}$ is a random vector with pdf $f(\cdot;\mathbf{u})$ belonging to some parametric family $\{f(\cdot;\mathbf{v}), \mathbf{v} \in \mathscr{V}\}$, and $\{S(\mathbf{X}) \geqslant \gamma\}$ is assumed to be a rare event. We can estimate ℓ using the likelihood ratio estimator (see also (5.59))

$$\widehat{\ell} = \frac{1}{N}\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}}\, W(\mathbf{X}_k;\mathbf{u},\mathbf{v}) \,, \tag{8.2}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from $f(\mathbf{x};\mathbf{v})$ and $W(\mathbf{X}_k;\mathbf{u},\mathbf{v}) = f(\mathbf{X}_k;\mathbf{u})/f(\mathbf{X}_k;\mathbf{v})$ is the likelihood ratio.

■ **EXAMPLE 8.1 Stochastic Shortest Path**

Let us return to Example 5.14 (see also Example 5.1), where the objective is to efficiently estimate the probability ℓ that the shortest path from node A to node B in the network of Figure 8.1 has a length of at least γ. The random lengths $X_1, \ldots, X_5$ of the links are assumed to be independent and exponentially distributed with means $u_1, \ldots, u_5$, respectively.

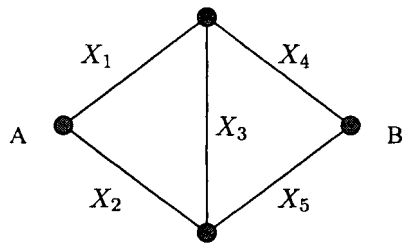


Figure 8.1 Shortest path from A to B.

Defining $\mathbf{X} = (X_1, \ldots, X_5)$, $\mathbf{u} = (u_1, \ldots, u_5)$ and

$$S(\mathbf{X}) = \min\{X_1 + X_4,\ X_1 + X_3 + X_5,\ X_2 + X_5,\ X_2 + X_3 + X_4\} \,,$$

the problem is cast in the framework of (8.1). As explained in Example 5.14, we can estimate (8.1) via (8.2) by drawing $X_1, \ldots, X_5$ independently from exponential distributions that are possibly *different* from the original ones. That is, $X_i \sim \mathsf{Exp}(v_i^{-1})$ instead of $X_i \sim \mathsf{Exp}(u_i^{-1})$, $i = 1, \ldots, 5$. The corresponding likelihood ratio was given in (5.73).

The challenging problem is how to select a vector $\mathbf{v} = (v_1, \ldots, v_5)$ that gives the most accurate estimate of ℓ for a given simulation effort. In the toy Example 5.14, this was achieved by first choosing the trial vector $\mathbf{w}$ equal to $\mathbf{u}$ and then applying the CE updating formula (5.69), possibly iterating the latter. This approach was possible because the event $\{S(\mathbf{x}) \geqslant \gamma\}$ was *not rare*. However, for the current problem (5.69) cannot be applied directly, since for rare events it returns, with high probability, the indeterminate expression $\frac{0}{0}$. To overcome this difficulty, we will use a different approach to selecting a good $\mathbf{v}$ by adopting a *two-stage* procedure where *both the level γ and the reference parameters $\mathbf{v}$ are updated.* One of the strengths of the CE method for rare-event simulation is that it provides a fast way to estimate accurately the optimal parameter vector $\mathbf{v}^*$.

Returning to the general situation, we have seen in Section 5.6 that for estimation problems of the form (8.1) the ideal (zero variance) importance sampling density is given by

$$g^*(\mathbf{x}) = \frac{f(\mathbf{x}; \mathbf{u})\, I_{\{S(\mathbf{x}) \geqslant \gamma\}}}{\ell} ,$$

which is the conditional pdf of $\mathbf{X}$ given $S(\mathbf{X}) \geqslant \gamma$. The idea behind the CE method is to get as close as possible to the optimal importance sampling distribution by using the Kullback–Leibler CE distance as a measure of closeness. Using a parametric class of densities $\{f(\mathbf{x}; \mathbf{v}), \mathbf{v} \in \mathscr{V}\}$, this means (see (5.61)) that the optimal reference parameter $\mathbf{v}^*$ is given by

$$\mathbf{v}^* = \operatorname*{argmax}_{\mathbf{v} \in \mathscr{V}} \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}} \ln f(\mathbf{X}; \mathbf{v})] . \tag{8.3}$$

We can, in principle, estimate $\mathbf{v}^*$ as

$$\operatorname*{argmax}_{\mathbf{v} \in \mathscr{V}} \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} \ln f(\mathbf{X}_k; \mathbf{v}) , \tag{8.4}$$

with $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim f(\cdot; \mathbf{u})$ — that is, using the stochastic counterpart of (8.3). However, as mentioned in Example 8.1, this is void of meaning if $\{S(\mathbf{X}) \geqslant \gamma\}$ is a rare event under $f(\cdot; \mathbf{u})$, since then most likely all indicators in the sum above will be zero.

To circumvent this problem we shall use a multilevel approach where we generate a sequence of reference parameters $\{\mathbf{v}_t,\ t \geqslant 0\}$ and a sequence of levels $\{\gamma_t,\ t \geqslant 1\}$ while iterating in both γ_t and $\mathbf{v}_t$. Our ultimate goal is to have $\mathbf{v}_t$ close to $\mathbf{v}^*$ after some number of iterations and to use $\mathbf{v}_t$ in the importance sampling density $f(\cdot; \mathbf{v}_t)$ to estimate ℓ.

We start with $\mathbf{v}_0 = \mathbf{u}$. Let ϱ be a not too small number, say $10^{-2} \leqslant \varrho \leqslant 10^{-1}$. In the first iteration, we choose $\mathbf{v}_1$ to be the optimal parameter for estimating $\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma_1)$, where γ_1 is the $(1 - \varrho)$-quantile of $S(\mathbf{X})$. That is, γ_1 is the largest real number for which

$$\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma_1) \geqslant \varrho .$$

Thus, if we simulate under $\mathbf{u}$, then level γ_1 is reached with a reasonably high probability of around ϱ. This enables us to estimate both γ_1 and $\mathbf{v}_1$ via Monte Carlo simulation. Namely, we can estimate γ_1 from a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\cdot; \mathbf{u})$ as follows. Calculate the performances $S(\mathbf{X}_i)$ for all i, and order them from smallest to largest: $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. Then γ_1 is estimated via the sample $(1 - \varrho)$-quantile $\widehat{\gamma}_1 = S_{(\lceil (1-\varrho)N \rceil)}$, where $\lceil a \rceil$ denotes the smallest integer larger than or equal to a (the so-called *ceiling* of a). The reference parameter $\mathbf{v}_1$ can be estimated via (8.4), replacing γ with the estimate of γ_1. Note that we can use here the *same* sample for estimating both $\mathbf{v}_1$ and γ_1. This means that $\mathbf{v}_1$ is

estimated on the basis of the $\lceil \varrho N \rceil$ best samples, that is, the samples $\mathbf{X}_i$ for which $S(\mathbf{X}_i)$ is greater than or equal to $\widehat{\gamma}_1$. These form the *elite samples* in the first iteration; let N^{e} denote the number of elite samples.

In the subsequent iterations we repeat these steps. Thus, we have the following two updating phases, starting from $\mathbf{v}_0 = \widehat{\mathbf{v}}_0 = \mathbf{u}$:

1. **Adaptive updating of γ_t.** For a fixed $\mathbf{v}_{t-1}$, let γ_t be the $(1-\varrho)$-quantile of $S(\mathbf{X})$ under $\mathbf{v}_{t-1}$. To estimate γ_t, draw a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\cdot; \widehat{\mathbf{v}}_{t-1})$ and evaluate the sample $(1-\varrho)$-quantile $\widehat{\gamma}_t$.

2. **Adaptive updating of $\mathbf{v}_t$.** For fixed γ_t and $\mathbf{v}_{t-1}$, derive $\mathbf{v}_t$ as

$$\mathbf{v}_t = \operatorname*{argmax}_{\mathbf{v} \in \mathscr{V}} \mathbb{E}_{\mathbf{v}_{t-1}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma_t\}} W(\mathbf{X}; \mathbf{u}, \mathbf{v}_{t-1}) \ln f(\mathbf{X}; \mathbf{v}) \right] . \tag{8.5}$$

   The stochastic counterpart of (8.5) is as follows: for fixed $\widehat{\gamma}_t$ and $\widehat{\mathbf{v}}_{t-1}$, derive $\widehat{\mathbf{v}}_t$ as the solution

$$\widehat{\mathbf{v}}_t = \operatorname*{argmax}_{\mathbf{v} \in \mathscr{V}} \frac{1}{N} \sum_{\mathbf{X}_k \in \mathcal{E}_t} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_{t-1}) \ln f(\mathbf{X}_k; \mathbf{v}) , \tag{8.6}$$

   where $\mathcal{E}_t$ is the *set of elite samples* in the t-th iteration, that is, the samples $\mathbf{X}_k$ for which $S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t$.

The procedure terminates when at some iteration T a level $\widehat{\gamma}_T$ is reached that is at least γ and thus the original value of γ can be used without getting too few samples. We then reset $\widehat{\gamma}_T$ to γ, reset the corresponding elite set, and deliver the final reference parameter $\widehat{\mathbf{v}}^*$, again using (8.6). This $\widehat{\mathbf{v}}^*$ is then used in (8.2) to estimate ℓ.

The resulting CE algorithm for rare-event probability estimation can thus be written as follows.

Algorithm 8.2.1 (Main CE Algorithm for Rare-Event Estimation)

1. *Define* $\widehat{\mathbf{v}}_0 = \mathbf{u}$. *Let* $N^{\mathrm{e}} = \lceil (1-\varrho)N \rceil$. *Set* $t = 1$ *(iteration counter).*

2. *Generate a random sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *according to the pdf* $f(\cdot; \widehat{\mathbf{v}}_{t-1})$. *Calculate the performances* $S(\mathbf{X}_i)$ *for all* i, *and order them from smallest to largest,* $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. *Let* $\widehat{\gamma}_t$ *be the sample* $(1-\varrho)$*-quantile of performances:* $\widehat{\gamma}_t = S_{(N^{\mathrm{e}})}$. *If* $\widehat{\gamma}_t > \gamma$, *reset* $\widehat{\gamma}_t$ *to* γ.

3. *Use the* **same** *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *to solve the stochastic program* (8.6).

4. *If* $\widehat{\gamma}_t < \gamma$, *set* $t = t + 1$ *and reiterate from Step 2. Otherwise, proceed with Step 5.*

5. *Let* T *be the final iteration counter. Generate a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_{N_1}$ *according to the pdf* $f(\cdot; \widehat{\mathbf{v}}_T)$ *and estimate* ℓ *via* (8.2).

Remark 8.2.1 In typical applications the sample size N in Steps 2–4 can be chosen smaller than the final sample size N_1 in Step 5.

Note that Algorithm 8.2.1 breaks down the complex problem of estimating the very small probability ℓ into a sequence of simple problems, generating a sequence of pairs $\{(\widehat{\gamma}_t, \widehat{\mathbf{v}}_t)\}$,

depending on ϱ, which is called the *rarity parameter*. Convergence of Algorithm 8.2.1 is discussed in [31]. Other convergence proofs for the CE method may be found in [21] and [6].

Remark 8.2.2 (Maximum Likelihood Estimator) Optimization problems of the form (8.6) appear frequently in statistics. In particular, if the W term is omitted — which will turn out to be important in CE optimization — one can also write (8.6) as

$$\widehat{\mathbf{v}}_t = \operatorname*{argmax}_{\mathbf{v}} \prod_{\mathbf{X}_k \in \mathcal{E}_t} f(\mathbf{X}_k; \mathbf{v}) ,$$

where the product is the joint density of the elite samples. Consequently, $\widehat{\mathbf{v}}_t$ is chosen such that the joint density of the elite samples is maximized. Viewed as a function of the parameter $\mathbf{v}$, rather than of the data $\{\mathcal{E}_t\}$, this joint density is called the *likelihood*. In other words, $\widehat{\mathbf{v}}_t$ is the *maximum likelihood estimator* (it maximizes the likelihood) of $\mathbf{v}$ based on the elite samples. When the W term is present, the form of the updating formula remains similar. Recall from Section 5.6 that for exponential families the updating rules for $\widehat{\mathbf{v}}_t$ can be obtained analytically; see also Section A.3 of the Appendix.

To gain a better understanding of the CE algorithm, we also present its *deterministic* version.

Algorithm 8.2.2 (Deterministic Version of the CE Algorithm)

1. *Define* $\mathbf{v}_0 = \mathbf{u}$. *Set* $t = 1$.

2. *Calculate* γ_t *as*

$$\gamma_t = \max \{s : \mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geqslant s) \geqslant \varrho\} . \tag{8.7}$$

   If $\gamma_t > \gamma$, *reset* γ_t *to* γ.

3. *Calculate* $\mathbf{v}_t$ *(see* (8.5)*) as*

$$\mathbf{v}_t = \operatorname*{argmax}_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma_t\}} W(\mathbf{X}; \mathbf{u}, \mathbf{v}_{t-1}) \ln f(\mathbf{X}; \mathbf{v}) \right] . \tag{8.8}$$

4. *If* $\gamma_t = \gamma$ **stop**; *otherwise, set* $t = t + 1$ *and repeat from Step 2.*

Note that, when compared with Algorithm 8.2.1, Step 5 is redundant in Algorithm 8.2.2.

To provide further insight into Algorithm 8.2.1, we shall follow it step by step in a number of toy examples.

■ **EXAMPLE 8.2 Exponential Distribution**

Let us revisit Examples 5.8, 5.10, and 5.12, where the goal was to estimate, via Monte Carlo simulation, the probability $\ell = \mathbb{P}_u(X \geqslant \gamma)$, with $X \sim \text{Exp}(u^{-1})$. Suppose that γ is large in comparison with u, so that $\ell = \mathrm{e}^{-\gamma/u}$ is a rare-event probability. The updating formula for $\widehat{v}_t$ in (8.6) follows from the optimization of

$$\sum_{X_k \in \mathcal{E}_t} W_k \ln \left(v^{-1} \mathrm{e}^{-X_k/v} \right) = - \sum_{X_k \in \mathcal{E}_t} W_k \ln v - \sum_{X_k \in \mathcal{E}_t} W_k \frac{X_k}{v} ,$$

where $W_k = \mathrm{e}^{-X_k(u^{-1}-v^{-1})}v/u$. To find the maximum of the right-hand side, we take derivatives and equate the result to 0:

$$-\sum_{X_k\in\mathcal{E}_t}\frac{W_k}{v}+\sum_{X_k\in\mathcal{E}_t}\frac{W_k X_k}{v^2}=0\,.$$

Solving this for v yields $\widehat{v}_t$. Thus,

$$\widehat{v}_t=\frac{\sum_{X_k\in\mathcal{E}_t}W_k X_k}{\sum_{X_k\in\mathcal{E}_t}W_k}\,. \tag{8.9}$$

In other words, $\widehat{v}_t$ is simply the sample mean of the elite samples weighted by the likelihood ratios. Note that without the weights $\{W_k\}$ we would simply have the maximum likelihood estimator of v for the $\mathrm{Exp}(v^{-1})$ distribution based on the elite samples in accordance with Remark 8.2.2. Note also that the updating formula (8.9) follows directly from (5.69). Similarly, the *deterministic* updating formula (8.8) gives

$$v_t=\frac{\mathbb{E}_{\mathbf{u}}\left[I_{\{X\geqslant\gamma_t\}}\,X\right]}{\mathbb{E}_{\mathbf{u}}\left[I_{\{X\geqslant\gamma_t\}}\right]}=\mathbb{E}_{\mathbf{u}}[X\mid X\geqslant\gamma_t]=u+\gamma_t\,,$$

where γ_t is the $(1-\varrho)$-quantile of the $\mathrm{Exp}(v_{t-1}^{-1})$ distribution. Thus, $\gamma_t=-v_{t-1}\ln\varrho$.

Assume for concreteness that $u=1$ and $\gamma=32$, which corresponds to $\ell\approx 1.2710^{-14}$. Table 8.1 presents the evolution of $\widehat{\gamma}_t$ and $\widehat{v}_t$ for $\varrho=0.05$ using sample size $N=1000$. Note that iteration $t=0$ corresponds to the original exponential pdf with expectation $u=1$, while iterations $t=1,2,3$ correspond to exponential pdfs with expectations $\widehat{v}_t$, $t=1,2,3$, respectively. Figure 8.2 illustrates the iterative procedure. We see that Algorithm 8.2.1 requires three iterations to reach the final level $\widehat{\gamma}_3=32$. In the third iteration the lowest value of the elite samples, $S_{(N^{\mathrm{e}})}$, was in fact greater than 32, so that in the final Step 2 of the algorithm we take $\widehat{\gamma}_3=\gamma=32$ instead. The corresponding reference parameter $\widehat{v}_3$ was found to be 32.82. Note that both parameters $\widehat{\gamma}_t$ and $\widehat{v}_t$ increase gradually, each time "blowing up" the tail of the exponential pdf.

The final step of Algorithm 8.2.1 now invokes the likelihood ratio estimator (8.2) to estimate ℓ, using a sample size N_1 that is typically larger than N.

Table 8.1 The evolution of $\widehat{\gamma}_t$ and $\widehat{v}_t$ for $\varrho=0.05$ with $\gamma=32$, using $N=1000$ samples.

t	$\widehat{\gamma}_t$	$\widehat{v}_t$
0	–	1
1	2.91	3.86
2	11.47	12.46
3	32	32.82

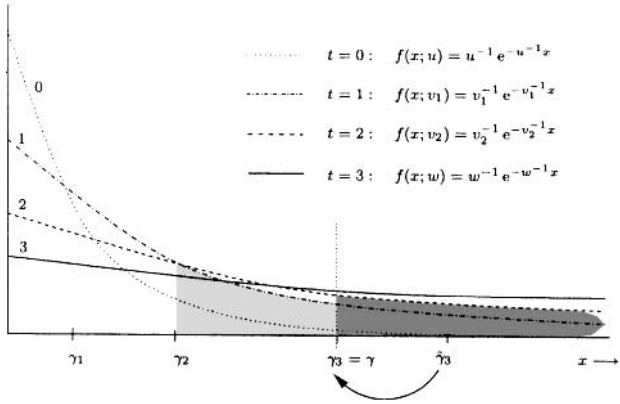


Figure 8.2 A three-level realization of Algorithm 8.2.1. Each shaded region has area ϱ.

■ EXAMPLE 8.3 Degeneracy

When γ is the maximum of $S(\mathbf{x})$, no "overshooting" of γ in Algorithm 8.2.1 occurs and therefore γ_t does not need to be reset. In such cases the sampling pdf may *degenerate* toward the *atomic* pdf that has all its mass concentrated at the points $\mathbf{x}$ where $S(\mathbf{x})$ is maximal. As an example, suppose we use a $\mathsf{Beta}(v, 1), v \geqslant 1$ family of importance sampling distributions, with nominal parameter $u = 1$ (corresponding to the uniform distribution), and take $S(X) = X$ and $\gamma = 1$. We find the updating formula for v from the optimization of

$$\sum_{X_k \in \mathscr{E}_k} W_k \ln(v X_k^{v-1}) = \sum_{X_k \in \mathscr{E}_k} W_k \ln v + \sum_{X_k \in \mathscr{E}_k} W_k (v-1) \ln X_k,$$

with $W_k = 1/(v X_k^{v-1})$. Hence,

$$\widehat{v}_t = \frac{\sum_{X_k \in \mathscr{E}_k} W_k}{-\sum_{X_k \in \mathscr{E}_k} W_k \ln X_k}.$$

Table 8.2 and Figure 8.3 show the evolution of parameters in the CE algorithm using $\varrho = 0.8$ and $N = 1000$. We see that $\widehat{\gamma}_t$ rapidly increases to γ and that the sampling pdf degenerates to the atomic density with mass at 1.

Table 8.2 The evolution of $\widehat{\gamma}_t$ and $\widehat{v}_t$ for the $\mathsf{Beta}(v, 1)$ example, with $\varrho = 0.8$ and $\gamma = 1$, using $N = 1000$ samples.

t	$\widehat{\gamma}_t$	$\widehat{v}_t$	t	$\widehat{\gamma}_t$	$\widehat{v}_t$
0	–	1	5	0.896	31.2
1	0.207	1.7	6	0.949	74.3
2	0.360	3.1	7	0.979	168.4
3	0.596	6.4	8	0.990	396.5
4	0.784	14.5	9	0.996	907.7

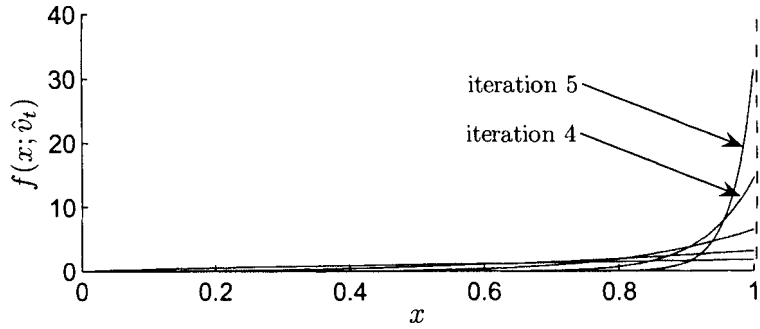


Figure 8.3 Degeneracy of the sampling distribution.

■ EXAMPLE 8.4 Coin Flipping

Consider the experiment where we flip n fair coins. We can describe this experiment via n independent Bernoulli random variables, $X_1, \ldots, X_n$, each with success parameter $1/2$. We write $\mathbf{X} = (X_1, \ldots, X_n) \sim \mathsf{Ber}(\mathbf{u})$, where $\mathbf{u} = (1/2, \ldots, 1/2)$ is the vector of success probabilities. Note that the range of $\mathbf{X}$ (the set of possible values it can take) contains 2^n elements. Suppose we are interested in estimating $\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma)$, with $S(\mathbf{X}) = \sum_{k=1}^{n} X_k$. We want to employ importance sampling using $\mathbf{X} \sim \mathsf{Ber}(\mathbf{p})$ for a possibly different parameter vector $\mathbf{p} = (p_1, \ldots, p_n)$. Consider two cases: (a) $\gamma = (n+1)/2$ (with n odd) and (b) $\gamma = n$. It is readily seen that for cases (a) and (b) the optimal importance sampling parameter vector is $\mathbf{p}^* = (1/2, \ldots, 1/2)$ and $\mathbf{p}^* = (1, \ldots, 1)$, respectively. The corresponding probabilities are $\ell = \frac{1}{2}$ and $\ell = \frac{1}{2^n}$, respectively. Note that in the first case ℓ is not a rare-event probability, but it is so for the second case (provided that n is large). Note also that in the second case $\mathsf{Ber}(\mathbf{p}^*)$ corresponds to a *degenerated* distribution that places all probability mass at the point $(1, 1, \ldots, 1)$.

Since $\{\mathsf{Ber}(\mathbf{p})\}$ forms an exponential family that is parameterized by the mean, it immediately follows from (5.69) that the updating formula for $\mathbf{p}$ in Algorithm 8.2.1 at the t-th iteration coincides with (8.9) and is given by

$$\widehat{p}_{t,i} = \frac{\sum_{\mathbf{X}_k \in \mathcal{E}_t} W_k X_{ki}}{\sum_{\mathbf{X}_k \mathcal{E}_t} W_k}, \quad i = 1, \ldots, n, \tag{8.10}$$

where X_{ki} is the i-th component of the k-th sample vector $\mathbf{X}_k \sim \mathsf{Ber}(\widehat{\mathbf{p}}_{t-1})$, and W_k is the corresponding likelihood ratio:

$$W_k = \prod_{i=1}^{n} q_i^{X_{ki}} r_i^{1-X_{ki}},$$

with $q_i = p_{0,i}/\widehat{p}_{t-1,i}$ and $r_i = (1 - p_{0,i})/(1 - \widehat{p}_{t-1,i})$, $i = 1, \ldots, n$. Thus, the i-th probability is updated as a weighted average of the number of 1s in the i-th position over all vectors in the elite sample.

As we shall see below, this simple coin flipping example will shed light on how rare-events estimation is connected with combinatorial optimization.

Remark 8.2.3 It is important to note that if we employ the deterministic CE algorithm 8.2.2 to any rare-event-type problem where the underlying distributions have finite supports *without fixing γ in advance*, it will iterate until it reaches some γ, denoted as γ_* (not necessarily the true optimal γ^*), and then stop. The corresponding importance sampling pdf $f(\mathbf{x}; \mathbf{v}_*)$ will be *degenerated*. For the above coin flipping example we will typically have in case (b) that $\gamma_* = \gamma^* = n$. The main Algorithm 8.2.1 behaves similarly, but in a stochastic rather than a deterministic sense. More precisely, for pdfs with finite supports and γ not fixed in advance, it will generate a tuple $(\widehat{\gamma}_T, \widehat{\mathbf{v}}_T)$ with $f(\mathbf{x}; \widehat{\mathbf{v}}_T)$ corresponding again typically to a degenerate pdf. This property of Algorithms 8.2.2 and 8.2.1 will be of crucial importance when dealing with combinatorial optimization problems in the next section. As mentioned, a combinatorial optimization problem can be viewed as a rare-event estimation problem in the sense that its optimal importance sampling pdf $f(\mathbf{x}; \mathbf{v}^*)$ is a degenerated one and coincides with the one generated by the deterministic rare-event Algorithm 8.2.2, provided that it keeps iterating in γ without fixing it in advance.

In the next example, we illustrate the behavior of Algorithm 8.2.1 when applied to a typical static simulation problem of estimating $\ell = \mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$. Note that the likelihood ratio estimator $\widehat{\ell}$ of ℓ in (8.2) is of the form $\widehat{\ell} = N^{-1} \sum_{k=1}^{n} Z_k$. We measure the efficiency of the estimator by its relative error (RE), which (recall (4.10)) is defined as

$$\mathrm{RE} = \mathrm{Var}(\widehat{\ell})^{1/2} / \mathbb{E}[\widehat{\ell}]$$

and which is estimated by $S/(\widehat{\ell}\sqrt{N})$, with

$$S^2 = N^{-1} \sum_{k=1}^{N} Z_k^2 - (\widehat{\ell})^2$$

being the sample variance of the $\{Z_i\}$. Assuming asymptotic normality of the estimator, the confidence intervals now follow in a standard way. For example, a 95% relative confidence interval for ℓ is given by

$$\widehat{\ell} \pm 1.96\, \widehat{\ell}\, \mathrm{RE}\,.$$

■ **EXAMPLE 8.5 Stochastic Shortest Path: Example 8.1 (Continued)**

Consider again the stochastic shortest path graph of Figure 8.1. Let us take the same nominal parameter vector $\mathbf{u}$ as in Example 5.14, that is, $\mathbf{u} = (1, 1, 0.3, 0.2, 0.1)$, and estimate the probability ℓ that the minimum path length is greater than $\gamma = 6$. Note that in Example 5.14 $\gamma = 1.5$ is used, which gives rise to an event that is not rare.

Crude Monte Carlo (CMC), with 10^8 samples — a very large simulation effort — gave an estimate $8.01 \cdot 10^{-6}$ with an estimated relative error of 0.035.

To apply Algorithm 8.2.1 to this problem, we need to establish the updating rule for the reference parameter $\mathbf{v} = (v_1, \ldots, v_5)$. Since the components $X_1, \ldots, X_5$ are independent and form an exponential family parameterized by the mean, this updating formula follows immediately from (5.69), that is,

$$\widehat{v}_{t,i} = \frac{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_{t-1})\, X_{ki}}{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_{t-1})}, \quad i = 1, \ldots, 5\,, \tag{8.11}$$

with $W(\mathbf{X}; \mathbf{u}, \mathbf{v})$ given in (5.73).

We take in all our experiments with Algorithm 8.2.1 the rarity parameter $\varrho = 0.1$, the sample size for Steps 2–4 of the algorithm $N = 10^3$, and for the final sample size $N_1 = 10^5$. Table 8.3 displays the results of Steps 1–4 of the CE algorithm. We see that after five iterations level $\gamma = 6$ is reached.

Table 8.3 Convergence of the sequence $\{(\widehat{\gamma}_t, \widehat{\mathbf{v}}_t)\}$.

t	$\widehat{\gamma}_t$	$\widehat{\mathbf{v}}_t$				
0		1.0000	1.0000	0.3000	0.2000	0.1000
1	1.1656	1.9805	2.0078	0.3256	0.2487	0.1249
2	2.1545	2.8575	3.0006	0.2554	0.2122	0.0908
3	3.1116	3.7813	4.0858	0.3017	0.1963	0.0764
4	4.6290	5.2803	5.6542	0.2510	0.1951	0.0588
5	6.0000	6.7950	6.7094	0.2882	0.1512	0.1360

Using the estimated optimal parameter vector of $\widehat{\mathbf{v}}_5 = (6.7950, 6.7094, 0.2882, 0.1512, 0.1360)$, Step 5 of the CE algorithm gave an estimate of $7.85 \cdot 10^{-6}$, with an estimated relative error of 0.035 — the same as for the CMC method with 10^8 samples. However, whereas the CMC method required more than an hour of computation time, the CE algorithm was finished in only one second, using a Matlab implementation on a 1500 MHz computer. We see that with a minimal amount of work, we have achieved a dramatic reduction of the simulation effort.

Table 8.4 presents the performance of Algorithm 8.2.1 for the above stochastic shortest path model, where instead of the exponential random variables we used Weib(α_i, λ_i) random variables, with $\alpha_i = 0.2$ and $\lambda_i = u_i^{-1}$, $i = 1, \ldots, 5$, where the $\{u_i\}$ are the same as before, that is, $\mathbf{u} = (1, 1, 0.3, 0.2, 0.1)$.

Table 8.4 The evolution of $\widehat{\mathbf{v}}_t$ for estimating the optimal parameter $\mathbf{v}^*$ with the TLR method and $\alpha = 0.2$. The estimated probability is $\widehat{\ell} = 3.30 \cdot 10^{-6}$, RE = 0.03.

t	$\widehat{\gamma}_t$	$\widehat{v}_{1t}$	$\widehat{v}_{2t}$	$\widehat{v}_{3t}$	$\widehat{v}_{4t}$	$\widehat{v}_{5t}$
0		1	1	1	1	1
1	3.633	2.0367	2.1279	0.9389	1.3834	1.4624
2	100.0	3.2690	3.3981	1.1454	1.3674	1.2939
3	805.3	4.8085	4.7221	0.9660	1.1143	0.9244
4	5720	6.6789	6.7252	0.6979	0.9749	1.0118
5	10000	7.5876	7.8139	1.0720	1.3152	1.2252

The Weibull distribution with shape parameter α less than 1 is an example of a *heavy-tailed* distribution. We use the TLR method (see Section 5.8) to estimate ℓ for $\gamma = 10{,}000$. Specifically, we first write (see (5.98)) $X_k = u_k Z_k^{1/\alpha}$, with $Z_k \sim \mathsf{Exp}(1)$, and then use importance sampling on the $\{Z_k\}$, changing the mean of Z_k from 1 to v_k, $k = 1, \ldots, 5$. The corresponding updating formula is again of the form (8.11), namely,

$$\widehat{v}_{t,i} = \frac{\sum_{k=1}^N I_{\{\widetilde{S}(\mathbf{Z}_k) \geqslant \widehat{\gamma}_t\}} \widetilde{W}(\mathbf{Z}_k; \mathbf{1}, \widehat{\mathbf{v}}_{t-1})\, Z_{ki}}{\sum_{k=1}^N I_{\{\widetilde{S}(\mathbf{Z}_k) \geqslant \widehat{\gamma}_t\}} \widetilde{W}(\mathbf{Z}_k; \mathbf{1}, \widehat{\mathbf{v}}_{t-1})}, \quad i = 1, \ldots, 5\,,$$

with $\widetilde{W}(\mathbf{Z}; \mathbf{1}, \mathbf{v})$ the likelihood ratio, and $\widetilde{S}(\mathbf{Z}) = S(\mathbf{X})$. Note that the "nominal" parameter here is $\mathbf{1} = (1, 1, 1, 1, 1)$, rather than $(1, 1, 0.3, 0.2, 0.1)$. Instead of using the TLR method, one could use the standard CE method here, where the components are sampled from $\{\text{Weib}(\alpha, v_i^{-1})\}$ and the $\{v_i\}$ are updated adaptively. One would obtain results similar (estimate and relative error) to those for the TLR case. The TLR is a convenient and quite general tool for importance sampling simulation, but it does not provide additional variance reduction; see also Exercises 8.5 and 8.6.

8.2.1 The Root-Finding Problem

In many applications one needs to estimate, for given ℓ, the *root*, γ, of the nonlinear equation

$$\mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}] = \ell \tag{8.12}$$

rather than estimate ℓ itself. We call such a problem a *root-finding* problem.

An estimate of γ in (8.12) based on the sample equivalent of $\mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}]$ can be obtained, for example, via stochastic approximation; see Chapter 7 and [32]. Alternatively, one can obtain γ using the CE method. The aim is to find a good trial vector $\widehat{\mathbf{v}}_T$ such that γ can be estimated as the smallest number $\widehat{\gamma}$ such that

$$\frac{1}{N_1} \sum_{k=1}^{N_1} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_T) \leqslant \ell \,. \tag{8.13}$$

In particular our main Algorithm 8.2.1 can be modified as follows.

Algorithm 8.2.3 (Root-Finding Algorithm)

1. *Define* $\widehat{\mathbf{v}}_0 = \mathbf{u}$, $N^{\mathrm{e}} = \lceil (1 - \varrho) N \rceil$. *Set* $t = 1$.
2. *Generate a random sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *from the density* $f(\cdot; \widehat{\mathbf{v}}_{t-1})$.
3. *Calculate the performances* $S(\mathbf{X}_1), \ldots, S(\mathbf{X}_N)$. *Order the performances from smallest to largest:* $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. *Let* $\widehat{\gamma}_t = S_{(N^{\mathrm{e}})}$.
4. *Calculate* $\widehat{\ell}_t = \max\{\ell, \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{v}}_{t-1})\}$.
5. *Determine* $\widehat{\mathbf{v}}_t$ *via* (8.6) *using the* **same** *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$.
6. *If* $\widehat{\ell}_t = \ell$, *proceed to Step 7; otherwise, let* $t = t + 1$ *and reiterate from Step 2.*
7. *Estimate* γ *via* (8.13) *using a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_{N_1} \sim f(\cdot; \widehat{\mathbf{v}}_T)$, *where* T *is the final iteration number.*

8.2.2 The Screening Method for Rare Events

Here we show how the screening method, introduced in Section 5.9, works for estimating rare-event probabilities of the form

$$\ell = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}] \,,$$

where we assume, as in Section 5.9, that the components of $\mathbf{X}$ are independent, that each component is distributed according to a one-dimensional exponential family parameterized

by the mean, and that $S(\mathbf{x})$ (and hence $H(\mathbf{x}) = I_{\{S(\mathbf{x}) \geqslant \gamma\}}$) is monotonically increasing in each component of $\mathbf{x}$. In particular, we shall present a modification of the two-stage Algorithm 5.9.1.

As in Algorithm 5.9.1, the main idea of the first stage of our modified algorithm is to identify the bottleneck parameters *without involving the likelihood ratio*. One might wonder how this could be possible given the fact that the estimation of the rare-event probability ℓ is essentially based on likelihood ratios. The trick is to execute the first stage (the screening part) by replacing γ with some γ_0 such that $\ell_0 = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma_0)$ *is not* a rare-event probability, say $10^{-2} \leqslant \ell_0 \leqslant 10^{-1}$. As soon as γ_0 is determined, the execution of the first stage is similar to the one in Algorithm 5.9.1. It reduces to finding the estimator, say $\widehat{\mathbf{v}}_0$, of the optimal parameter vector $\mathbf{v}_0^*$ obtained from (8.4), where γ is replaced by γ_0. Note that (8.4) does not contain the likelihood ratio term $W(\mathbf{X}; \mathbf{u}, \mathbf{w})$. It is important to note again that the components of $\mathbf{v}_0^*$ are at least as large as the corresponding elements of $\mathbf{u}$, and thus we can classify the bottleneck and nonbottleneck parameters according to the relative perturbation

$$\delta_i = \frac{\widehat{v}_i - u_i}{u_i}, \; i = 1, \ldots, n \,,$$

which is the core of the screening algorithm.

Below we present the modified version of the two-stage screening CE-SCR Algorithm 5.9.1 suitable for rare events. We use the same notation as in Section 5.9.1.

Algorithm 8.2.4 (Two-Stage Screening CE-SCR Algorithm)

1. *Initialize the set of bottleneck elements to* $B_0 = \{1, \ldots, n\}$. *Set* $t = 1$.
2. *Generate a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *from* $f(\mathbf{x}; \mathbf{u})$ *and compute* $\widehat{\gamma}_0$, *the* $(1 - \varrho)$- *sample quantile of the sample performances* $\{S(\mathbf{X}_i)\}$.
3. *Generate a different sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *from* $f(\mathbf{x}; \mathbf{u})$ *and deliver the CE solution of the stochastic program (5.107), with* $\gamma = \gamma_0$. *Denote the solution by* $\widehat{\mathbf{v}}_t = (\widehat{v}_{t1}, \ldots, \widehat{v}_{tn})$. *Note that* $\widehat{\mathbf{v}}_t$ *is an* n*-dimensional parameter vector.*
4. *Calculate the relative perturbation for each element* $\widehat{v}_{ti}$, $i = 1, \ldots, n$ *as*

$$\delta_{ti} = \frac{\widehat{v}_{ti} - u_i}{u_i} \,. \tag{8.14}$$

5. *If* $\delta_{ti} < \delta$, *where* δ *is some threshold value, say* $\delta = 0.1$ *(note that a negative* δ_{ti} *automatically satisfies* $\delta_{ti} < \delta$*), set* $\widehat{v}_{ti} = u_i$, *that is, identify the* i*-th element of the vector* $\mathbf{v}$ *as a nonbottleneck parameter. Otherwise, identify it as a bottleneck one. Let* B_t *be the set of bottleneck elements at iteration* t.
6. *Repeat Steps 3–5 several times, say* d *times, increasing each time* t *by 1 and updating the set* B_t. *Note that the sequence of sets* B_t, $t = 1, \ldots, d$ *is nonincreasing.*
7. *Apply the standard CE algorithm to estimate the optimal parameter vector* $\mathbf{v}_B$, *with* $B = B_d$. *Deliver (5.106), that is,*

$$\widehat{\ell}_B = \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} \, W_B(\mathbf{X}_{kB}; \mathbf{u}_B, \widehat{\mathbf{v}}_B),$$

as the resulting estimator of the rare-event probability ℓ.

8.2.2.1 Numerical Results Next, we present numerical studies with Algorithm 8.2.4 for the $m \times n$ bridge system in Figure 5.5 on page 156. We are interested in estimating the rare-event probability $\ell = \mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$ that the length $S(\mathbf{X})$ of the shortest path through the graph is greater than or equal to γ, where

$$S(\mathbf{X}) = \min\{Y_{11} + \cdots + Y_{1n}, \ldots, Y_{m1} + \cdots + Y_{mn}\}$$

and the Y_{ij} are defined in (5.109). Note that the operator "max" in (5.110) is replaced by "min" here. The random variables X_{ijk} are assumed to be $\mathsf{Exp}(u_{ijk})$ distributed. As in the numerical example in Section 5.9.1, it is important to realize that the $\{X_{ijk}\}$ are *not* parameterized by the mean, and that one needs to take instead $1/u_{ijk}$ and $1/\widehat{v}_{ijk}$ to compare the relative perturbations as described above. For the same reason, the parameter values corresponding to the bottleneck elements should be *smaller* than those for the nonbottleneck ones. As in Section 5.9, we purposely select (in advance) some elements of our model to be bottlenecks.

Table 8.5 presents the performance of Algorithm 8.2.4 for the 2×2 model with eight bottlenecks, using $\delta = 0.1$, $\gamma = 6$ and the sample sizes $N = 50{,}000$ and $N_1 = 500{,}000$. In particular, we set the bottleneck parameters u_{111}, u_{112}, u_{121}, u_{122}, u_{211}, u_{212}, u_{221}, u_{222} to 1 and the remaining 12 elements to 4.

Table 8.5 Performance of Algorithm 8.2.4 for the 2×2 model. We set $\delta = 0.1$, $\gamma = 6$, $N = 50{,}000$, $N_1 = 500{,}000$.

	CE	VM	CE-SCR	VM-SCR
Mean $\widehat{\ell}$	2.92E-8	2.96E-8	2.88E-8	2.81E-8
Max $\widehat{\ell}$	3.93E-8	3.69E-8	3.56E-8	3.29E-8
Min $\widehat{\ell}$	2.46E-8	2.65E-8	2.54E-8	2.45E-8
RE	0.166	0.102	0.109	0.077
CPU	6.03	9.31	6.56	9.12

From the results of Table 8.5 it follows that, for this relatively small model, both CE and VM perform similarly to their screening counterparts. We will see further on that as the complexity of the model increases, VM-SCR outperforms its three alternatives and, in particular, CE-SCR.

Table 8.6 presents the typical dynamics for detecting the bottleneck parameters at the first stage of Algorithm 8.2.4 for the above 2×2 model with 20 parameters, 8 of which are bottlenecks. Similar to Table 5.5, in Table 8.6 the 0s and 1s indicate which parameters are detected as nonbottleneck and bottleneck ones, respectively, and t denotes the iteration number at the first stage of the algorithm.

Table 8.6 Typical dynamics for detecting the bottleneck parameters at the first stage of Algorithm 8.2.4.

t	u_{111}	u_{112}	u_{113}	u_{114}	u_{115}	u_{121}	u_{122}	u_{123}	u_{124}	u_{125}
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	1	0	0
2	1	1	0	0	0	1	1	0	0	0
3	1	1	0	0	0	1	1	0	0	0
4	1	1	0	0	0	1	1	0	0	0
5	1	1	0	0	0	1	1	0	0	0

t	u_{211}	u_{212}	u_{213}	u_{214}	u_{215}	u_{221}	u_{222}	u_{223}	u_{224}	u_{225}
0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	0	1	1	0	1	1
2	1	1	1	0	0	1	1	0	1	1
3	1	1	0	0	0	1	1	0	0	0
4	1	1	0	0	0	1	1	0	0	0
5	1	1	0	0	0	1	1	0	0	0

It is readily seen that after the first iteration we have 13 bottleneck parameters and after the second one 11 bottleneck parameters; after the third iteration the process stabilizes, delivering the 8 true bottleneck parameters.

Table 8.7 presents a typical evolution of the sequence $\{(\widehat{\gamma}_t, \widehat{\mathbf{v}}_t)\}$ for the elements of the above 2×2 model for the VM and VM-SCR methods. We see in this table that the bottleneck elements decrease more than three times, while the nonbottleneck elements fluctuate around their nominal values 4.

Table 8.7 Typical evolution of the sequence $\{\widehat{\mathbf{v}}_t\}$ for the VM and VM-SCR methods.

	VM					VM-SCR				
t	$\widehat{v}_{111}$	$\widehat{v}_{112}$	$\widehat{v}_{113}$	$\widehat{v}_{114}$	$\widehat{v}_{115}$	$\widehat{v}_{111}$	$\widehat{v}_{112}$	$\widehat{v}_{113}$	$\widehat{v}_{114}$	$\widehat{v}_{115}$
0	1.000	1.000	4.000	4.000	4.000	1.000	1.000	4	4	4
1	0.759	0.771	3.944	3.719	3.839	0.760	0.771	4	4	4
2	0.635	0.613	3.940	3.681	3.734	0.638	0.605	4	4	4
3	0.524	0.517	4.060	3.297	3.608	0.506	0.491	4	4	4
4	0.443	0.415	3.370	3.353	3.909	0.486	0.447	4	4	4
5	0.334	0.332	3.689	3.965	4.250	0.402	0.371	4	4	4
6	0.378	0.365	3.827	3.167	4.188	0.348	0.317	4	4	4
7	0.357	0.358	3.881	4.235	4.929	0.375	0.347	4	4	4
8	0.285	0.271	4.011	2.982	4.194	0.285	0.298	4	4	4
9	0.287	0.301	3.249	2.879	3.409	0.288	0.254	4	4	4

We conclude with a larger model, namely a 3×10 model, in which u_{111}, u_{112}, $u_{211}, u_{212}, u_{311}$ and u_{312} are chosen as bottleneck parameters and are set to 1, while the remaining parameters are set to 4. Table 8.8 presents the performance of Algorithm 8.2.4

for this model using $\delta = 0.1$, $\gamma = 6$, and $N = N_1 = 400{,}000$. In this case, both CE and VM found the true six bottlenecks. Note that VM-SCR is the most accurate of the three alternatives and that CE underestimates ℓ. Thus, for this relatively large model, CE without screening is affected by the degeneracy of the likelihood ratio, presenting a product of 150 terms.

Table 8.8 Performance of Algorithm 8.2.4 for the 3×10 model with six bottlenecks. We set $\delta = 0.1$, $\gamma = 6$, $N = 400{,}000$, $N_1 = 400{,}000$.

	CE	VM	CE-SCR	VM-SCR
Mean $\widehat{\ell}$	2.44E-8	5.34E-8	5.28E-8	5.17E-8
Max $\widehat{\ell}$	5.82E-8	7.18E-8	8.34E-8	6.93E-8
Min $\widehat{\ell}$	4.14E-15	2.76E-8	2.74E-8	4.32E-8
RE	1.05	0.28	0.33	0.15
CPU	247	482	303	531

8.3 THE CE METHOD FOR OPTIMIZATION

In this section we show how the CE method works for optimization. Suppose we wish to maximize a function $S(\mathbf{x})$ over some set $\mathcal{X}$. Let us denote the maximum by γ^*; thus,

$$\gamma^* = \max_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x}) \,. \tag{8.15}$$

The problem is called a *discrete* or *continuous* optimization problem, depending on whether $\mathcal{X}$ is discrete or continuous. An optimization problem involving both discrete and continuous variables is called a *mixed* optimization problem. A discrete optimization problem is sometimes called a *combinatorial optimization problem*, which is the main focus of this section.

The CE method takes a novel approach to optimization problems by casting the original problem (8.15) into an *estimation problem of rare-event probabilities*. By doing so, the CE method aims to locate an optimal parametric sampling distribution, that is, a probability distribution on $\mathcal{X}$, rather than locating the optimal solution directly. To this end, we define a collection of indicator functions $\{I_{\{S(\mathbf{x}) \geqslant \gamma\}}\}$ on $\mathcal{X}$ for various levels $\gamma \in \mathbb{R}$. Next, let $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ be a family of probability densities on $\mathcal{X}$ parameterized by a real-valued parameter vector $\mathbf{v}$. For a fixed $\mathbf{u} \in \mathcal{V}$ we associate with (8.15) the problem of estimating the rare-event probability

$$\ell(\gamma) = \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}_{\mathbf{u}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma\}} \right], \tag{8.16}$$

where $\mathbb{P}_{\mathbf{u}}$ is the probability measure under which the random state $\mathbf{X}$ has a discrete pdf $f(\cdot; \mathbf{u})$ and $\mathbb{E}_{\mathbf{u}}$ denotes the corresponding expectation operator. We call the estimation problem (8.16) the *associated stochastic problem*.

It is crucial to understand that one of the main goals of CE in optimization is to generate a sequence of pdfs $f(\cdot; \widehat{\mathbf{v}}_0), f(\cdot; \widehat{\mathbf{v}}_1), \ldots$ converging to a degenerate measure (Dirac measure) that assigns all probability mass to a single state $\mathbf{x}_T$, for which, by definition, the function value is either optimal or very close to it.

As soon as the associated stochastic problem is defined, we approximate the optimal solution, say $\mathbf{x}^*$, of (8.15) by applying Algorithm 8.2.1 for rare-event estimation, but without fixing γ in advance. It is plausible that if $\widehat{\gamma}^*$ is close to γ^*, then $f(\cdot; \widehat{\mathbf{v}}_T)$ assigns most of its probability mass close to $\mathbf{x}^*$. Thus, any $\mathbf{X}$ drawn from this distribution can be used as an approximation to the optimal solution $\mathbf{x}^*$ and the corresponding function value as an approximation to the true optimal γ^* in (8.15).

To provide more insight into the relation between combinatorial optimization and rare-event estimation, we first revisit the coin flipping problem of Example 8.4, but from an optimization rather than an estimation perspective. This will serve as a highlight to *all real combinatorial optimization problems*, such as the maximal cut problem and the TSP considered in the next section, in the sense that only the sample function $S(\mathbf{X})$ and the trajectory generation algorithm will be different from the toy example below, while the updating of the sequence $\{(\gamma_t, \mathbf{v}_t)\}$ will always be determined from the *same* principles.

■ **EXAMPLE 8.6 Flipping n Coins: Example 8.4 Continued**

Suppose we want to maximize

$$S(\mathbf{x}) = \sum_{i=1}^{n} x_i \,,$$

where $x_i = 0$ or 1 for all $i = 1, \ldots, n$. Clearly, the optimal solution to (8.15) is $\mathbf{x}^* = (1, \ldots, 1)$. The simplest way to put the deterministic program (8.15) into a stochastic framework is to associate with each component x_i, $i = 1, \ldots, n$ a Bernoulli random variable X_i, $i = 1, \ldots, n$. For simplicity, assume that all $\{X_i\}$ are independent and that each component i has success probability $1/2$. By doing so, the associated stochastic problem (8.16) becomes a rare-event estimation problem. Taking into account that there is a single solution $\mathbf{x}^* = (1, \ldots, 1)$, using the CMC method we obtain $\ell(\gamma^*) = 1/|\mathscr{X}|$, where $|\mathscr{X}| = 2^n$, which for large n is a very small probability. Instead of estimating $\ell(\gamma)$ via CMC, we can estimate it via importance sampling using $X_i \sim \mathsf{Ber}(p_i)$, $i = 1, \ldots, n$.

The next step is, clearly, to apply Algorithm 8.2.1 to (8.16) without fixing γ in advance. As mentioned in Remark 8.2.3, CE Algorithm 8.2.1 should be viewed as the stochastic counterpart of the deterministic CE Algorithm 8.2.2, and the latter will iterate until it reaches a local maximum. We thus obtain a sequence $\{\widehat{\gamma}_t\}$ that converges to a local or global maximum, which can be taken as an estimate for the true optimal solution γ^*.

In summary, in order to solve a combinatorial optimization problem, we shall employ the CE Algorithm 8.2.1 for rare-event estimation without fixing γ in advance. By doing so, the CE algorithm for optimization can be viewed as a modified version of Algorithm 8.2.1. In particular, by analogy to Algorithm 8.2.1, we choose a not very small number ϱ, say $\varrho = 10^{-2}$, initialize the parameter vector $\mathbf{u}$ by setting $\mathbf{v}_0 = \mathbf{u}$, and proceed as follows.

1. **Adaptive updating of γ_t.** For a fixed $\mathbf{v}_{t-1}$, let γ_t be the $(1-\varrho)$-quantile of $S(\mathbf{X})$ under $\mathbf{v}_{t-1}$. As before, an estimator $\widehat{\gamma}_t$ of γ_t can be obtained by drawing a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\cdot; \mathbf{v}_{t-1})$ and then evaluating the sample $(1-\varrho)$-quantile of the performances as

$$\widehat{\gamma}_t = S_{(\lceil (1-\varrho)N \rceil)} \,. \tag{8.17}$$

2. **Adaptive updating of $\mathbf{v}_t$.** For fixed γ_t and $\mathbf{v}_{t-1}$, derive $\mathbf{v}_t$ from the solution of the program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}}[I_{\{S(\mathbf{X})\geqslant\gamma_t\}} \ln f(\mathbf{X};\mathbf{v})] \,. \tag{8.18}$$

The stochastic counterpart of (8.18) is as follows: for fixed $\widehat{\gamma}_t$ and $\widehat{\mathbf{v}}_{t-1}$, derive $\widehat{\mathbf{v}}_t$ from the following program:

$$\max_{\mathbf{v}} \widehat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k)\geqslant\widehat{\gamma}_t\}} \ln f(\mathbf{X}_k;\mathbf{v}) \,. \tag{8.19}$$

It is important to observe that in contrast to (8.5) and (8.6) (for the rare-event setting) (8.18) and (8.19) *do not contain the likelihood ratio terms W*. The reason is that in the rare-event setting the initial (nominal) parameter $\mathbf{u}$ is specified in advance and is an essential part of the estimation problem. In contrast, the initial reference vector $\mathbf{u}$ in the associated stochastic problem is quite arbitrary. In effect, by dropping the W term, we can efficiently estimate at each iteration t the CE optimal reference parameter vector $\mathbf{v}_t$ for the rare-event probability $\mathbb{P}_{\mathbf{v}_t}(S(\mathbf{X}) \geqslant \gamma_t) \geq \mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geqslant \gamma_t)$, even for high-dimensional problems.

Remark 8.3.1 (Smoothed Updating) Instead of updating the parameter vector $\mathbf{v}$ directly via the solution of (8.19), we use the following *smoothed* version

$$\widehat{\mathbf{v}}_t = \alpha\widetilde{\mathbf{v}}_t + (1-\alpha)\widehat{\mathbf{v}}_{t-1}, \tag{8.20}$$

where $\widetilde{\mathbf{v}}_t$ is the parameter vector obtained from the solution of (8.19) and α is called the *smoothing parameter*, where typically $0.7 < \alpha \leqslant 1$. Clearly, for $\alpha = 1$ we have our original updating rule. The reason for using the smoothed (8.20) instead of the original updating rule is twofold: (a) to smooth out the values of $\widehat{\mathbf{v}}_t$ and (b) to reduce the probability that some component $\widehat{v}_{t,i}$ of $\widehat{\mathbf{v}}_t$ will be 0 or 1 at the first few iterations. This is particularly important when $\widehat{\mathbf{v}}_t$ is a vector or matrix of *probabilities*. Note that for $0 < \alpha \leqslant 1$ we always have $\widehat{v}_{t,i} > 0$, while for $\alpha = 1$ we might have (even at the first iterations) $\widehat{v}_{t,i} = 0$ or $\widehat{v}_{t,i} = 1$ for some indices i. As result, the algorithm will converge to a wrong solution.

Thus, the main CE optimization algorithm, which includes smoothed updating of parameter vector $\mathbf{v}$ and which presents a slight modification of Algorithm 8.2.1 can be summarized as follows.

Algorithm 8.3.1 (Main CE Algorithm for Optimization)

1. *Choose an initial parameter vector $\mathbf{v}_0 = \widehat{\mathbf{v}}_0$. Set $t = 1$ (level counter).*

2. *Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from the density $f(\cdot;\mathbf{v}_{t-1})$ and compute the sample $(1-\varrho)$-quantile $\widehat{\gamma}_t$ of the performances according to (8.17).*

3. *Use the* **same** *sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ and solve the stochastic program (8.19). Denote the solution by $\widetilde{\mathbf{v}}_t$.*

4. *Apply (8.20) to smooth out the vector $\widetilde{\mathbf{v}}_t$.*

5. *If the stopping criterion is met, stop; otherwise, set $t = t + 1$, and return to Step 2.*

Remark 8.3.2 (Minimization) When $S(\mathbf{x})$ is to be *minimized* instead of maximized, we simply change the inequalities "$\geq$" to "$\leq$" and take the ϱ-quantile instead of the $(1-\varrho)$-quantile. Alternatively, we can just maximize $-S(\mathbf{x})$.

As a stopping criterion one can use, for example: if for some $t \geqslant d$, say $d = 5$,

$$\widehat{\gamma}_t = \widehat{\gamma}_{t-1} = \cdots = \widehat{\gamma}_{t-d} \,, \tag{8.21}$$

then stop. As an alternative estimate for γ^* one can consider

$$\widetilde{\gamma}_T = \max_{0 \leqslant s \leqslant T} \widehat{\gamma}_s \,. \tag{8.22}$$

Note that the initial vector $\widehat{\mathbf{v}}_0$, the sample size N, the stopping parameter d, and the number ϱ have to be specified in advance, but the rest of the algorithm is "self-tuning". Note also that, by analogy to the simulated annealing algorithm, γ_t may be viewed as the "annealing temperature". In contrast to simulated annealing, where the cooling scheme is chosen in advance, in the CE algorithm it is updated adaptively.

■ **EXAMPLE 8.7 Example 8.6 Continued: Flipping Coins**

In this case, the random vector $\mathbf{X} = (X_1, \ldots, X_n) \sim \mathrm{Ber}(\mathbf{p})$ and the parameter vector $\mathbf{v}$ is $\mathbf{p}$. Consequently, the pdf is

$$f(\mathbf{X}; \mathbf{p}) = \prod_{i=1}^{n} p_i^{X_i} (1-p_i)^{1-X_i} \,,$$

and since each X_i can only be 0 or 1,

$$\frac{\partial}{\partial p_i} \ln f(\mathbf{X}; \mathbf{p}) = \frac{X_i}{p_i} - \frac{1-X_i}{1-p_i} = \frac{1}{(1-p_i)p_i}(X_i - p_i) \,.$$

Now we can find the optimal parameter vector $\mathbf{p}$ of (8.19) by setting the first derivatives with respect to p_i equal to zero for $i = 1, \ldots, n$, that is,

$$\frac{\partial}{\partial p_i} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} \ln f(\mathbf{X}_k; \mathbf{p}) = \frac{1}{(1-p_i)\,p_i} \sum_{i=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} \,(X_{ki} - p_i) = 0 \,.$$

Thus, we obtain

$$p_i = \frac{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}} X_{ki}}{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \gamma\}}} \,, \tag{8.23}$$

which gives the same updating formula as (8.10) *except for the W term.* Recall that the updating formula (8.23) holds, in fact, for all one-dimensional exponential families that are parameterized by the mean; see (5.69). Note also that the parameters are simply updated via their maximum likelihood estimators, using only the elite samples; see Remark 8.2.2.

Algorithm 8.3.1 can, in principle, be applied to any discrete and continuous optimization problem. However, for each problem two essential actions need to be taken:

1. We need to specify how the samples are generated. In other words, we need to specify the family of densities $\{f(\cdot;\mathbf{v})\}$.
2. We need to update the parameter vector $\mathbf{v}$ based on CE minimization program (8.19), which is the *same* for all optimization problems.

In general, there are many ways to generate samples from $\mathscr{X}$, and it is not always immediately clear which method will yield better results or easier updating formulas.

Remark 8.3.3 (Parameter Selection) The choice of the sample size N and the rarity parameter ϱ depends on the size of the problem and the number of parameters in the associated stochastic problem. Typical choices are $\varrho = 0.1$ or $\varrho = 0.01$ and $N = c\,K$, where K is the number of parameters that need to be estimated/updated and c is a constant between 1 and 10.

By analogy to Algorithm 8.2.2 we also present the deterministic version of Algorithm 8.3.1, which will be used below.

Algorithm 8.3.2 (Deterministic CE Algorithm for Optimization)

1. *Choose some* $\mathbf{v}_0$. *Set* $t = 1$.
2. *Calculate* γ_t *as*
$$\gamma_t = \max\{s : \mathbb{P}_{\mathbf{v}_{t-1}}(S(\mathbf{X}) \geqslant s) \geqslant \varrho\}\,. \tag{8.24}$$
3. *Calculate* $\mathbf{v}_t$ *as*
$$\mathbf{v}_t = \operatorname*{argmax}_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}}[I_{\{S(\mathbf{X})\geqslant\gamma_t\}} \ln f(\mathbf{X};\mathbf{v})]\,. \tag{8.25}$$
4. *If for some* $t \geqslant d$, *say* $d = 5$,
$$\gamma_t = \gamma_{t-1} = \cdots = \gamma_{t-d}\,, \tag{8.26}$$
then **stop** *(let* T *denote the final iteration); otherwise, set* $t = t+1$ *and reiterate from Step 2.*

Remark 8.3.4 Note that instead of the CE distance we could minimize the variance of the estimator, as discussed in Section 5.6. As mentioned, the main reason for using CE is that for exponential families the parameters can be updated analytically, rather than numerically as for the VM procedure.

Below we present several applications of the CE method to combinatorial optimization, namely the max-cut, the bipartition and the TSP. We demonstrate numerically the efficiency of the CE method and its fast convergence for several case studies. For additional applications of CE see [31] and the list of references at the end of this chapter.

8.4 THE MAX-CUT PROBLEM

The maximal cut or *max-cut* problem can be formulated as follows. Given a graph $G = G(V,E)$ with a set of nodes $V = \{1,\ldots,n\}$ and a set of edges E between the nodes, partition the nodes of the graph into two arbitrary subsets V_1 and V_2 such that the sum of

the weights (costs) c_{ij} of the edges going from one subset to the other is maximized. Note that some of the c_{ij} may be 0 — indicating that there is, in fact, no edge from i to j.

As an example, consider the graph in Figure 8.4, with corresponding cost matrix $C = (c_{ij})$ given by

$$C = \begin{pmatrix} 0 & 2 & 2 & 5 & 0 \\ 2 & 0 & 1 & 0 & 3 \\ 2 & 1 & 0 & 4 & 2 \\ 5 & 0 & 4 & 0 & 1 \\ 0 & 3 & 2 & 1 & 0 \end{pmatrix} . \tag{8.27}$$

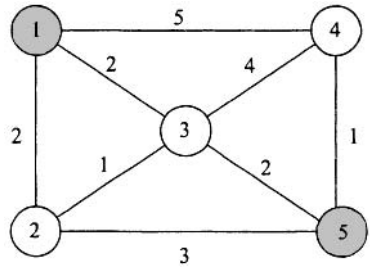


Figure 8.4 A six-node network with the cut $\{\{1,5\},\{2,3,4\}\}$.

Here the cut $\{\{1,5\},\{2,3,4\}\}$ has cost

$$c_{12} + c_{13} + c_{14} + c_{52} + c_{53} + c_{54} = 2 + 2 + 5 + 3 + 2 + 1 = 15 .$$

A cut can be conveniently represented via its corresponding *cut vector* $\mathbf{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ if node i belongs to same partition as 1 and 0 otherwise. For example, the cut in Figure 8.4 can be represented via the cut vector $(1,0,0,0,1)$. For each cut vector $\mathbf{x}$, let $\{V_1(\mathbf{x}), V_2(\mathbf{x})\}$ be the partition of V induced by $\mathbf{x}$, such that $V_1(\mathbf{x})$ contains the set of indices $\{i : x_i = 1\}$. If not stated otherwise, we set $x_1 = 1 \in V_1$.

Let $\mathscr{X}$ be the set of all cut vectors $\mathbf{x} = (1, x_2, \ldots, x_n)$ and let $S(\mathbf{x})$ be the corresponding cost of the cut. Then

$$S(\mathbf{x}) = \sum_{i \in V_1(\mathbf{x}),\, j \in V_2(\mathbf{x})} c_{ij} . \tag{8.28}$$

It is readily seen that the total number of cut vectors is

$$|\mathscr{X}| = 2^{n-1}. \tag{8.29}$$

We shall assume below that the graph is *undirected*. Note that for a *directed* graph the cost of a cut $\{V_1, V_2\}$ includes the cost of the edges both from V_1 to V_2 and from V_2 to V_1. In this case, the cost corresponding to a cut vector $\mathbf{x}$ is therefore

$$S(\mathbf{x}) = \sum_{i \in V_1(\mathbf{x}),\, j \in V_2(\mathbf{x})} (c_{ij} + c_{ji}) . \tag{8.30}$$

Next, we generate random cuts and update of the corresponding parameters using the CE Algorithm 8.3.1. The most natural and easiest way to generate the cut vectors is

to let $X_2, \ldots, X_n$ be independent Bernoulli random variables with success probabilities $p_2, \ldots, p_n$.

Algorithm 8.4.1 (Random Cuts Generation)

1. *Generate an n-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_n)$ from* $\mathsf{Ber}(\mathbf{p})$ *with independent components, where $\mathbf{p} = (1, p_2, \ldots, p_n)$.*

2. *Construct the partition $\{V_1(\mathbf{X}), V_2(\mathbf{X})\}$ of V and calculate the performance $S(\mathbf{X})$ as in* (8.28).

The updating formulas for $\widehat{p}_{t,i}$ are the same as for the toy Example 8.7 and are given in (8.23).

The following toy example illustrates, step by step, the workings of the deterministic CE Algorithm 8.3.2. The small size of the problem allows us to make all calculations analytically, that is, using directly the updating rules (8.24) and (8.25) rather than their stochastic counterparts.

■ **EXAMPLE 8.8 Illustration of Algorithm 8.3.2**

Consider the five-node graph presented in Figure 8.4. The 16 possible cut vectors (see (8.29)) and the corresponding cut values are given in Table 8.9.

Table 8.9 The 16 possible cut vectors of Example 8.8.

$\mathbf{X}$	V_1	V_2	$S(\mathbf{X})$
(1,0,0,0,0)	$\{1\}$	$\{2,3,4,5\}$	9
(1,1,0,0,0)	$\{1,2\}$	$\{3,4,5\}$	11
(1,0,1,0,0)	$\{1,3\}$	$\{2,4,5\}$	14
(1,0,0,1,0)	$\{1,4\}$	$\{2,3,5\}$	9
(1,0,0,0,1)	$\{1,5\}$	$\{2,3,4\}$	15
(1,1,1,0,0)	$\{1,2,3\}$	$\{4,5\}$	14
(1,1,0,1,0)	$\{1,2,4\}$	$\{3,5\}$	11
(1,1,0,0,1)	$\{1,2,5\}$	$\{3,4\}$	11
(1,0,1,1,0)	$\{1,3,4\}$	$\{2,5\}$	6
(1,0,1,0,1)	$\{1,3,5\}$	$\{2,4\}$	16
(1,0,0,1,1)	$\{1,4,5\}$	$\{2,3\}$	13
(1,1,1,1,0)	$\{1,2,3,4\}$	$\{5\}$	6
(1,1,1,0,1)	$\{1,2,3,5\}$	$\{4\}$	10
(1,1,0,1,1)	$\{1,2,4,5\}$	$\{3\}$	9
(1,0,1,1,1)	$\{1,3,4,5\}$	$\{2\}$	6
(1,1,1,1,1)	$\{1,2,3,4,5\}$	$\emptyset$	0

It follows that in this case the optimal cut vector is $\mathbf{x}^* = (1,0,1,0,1)$ with $S(\mathbf{x}^*) = \gamma^* = 16$.

We shall show next that in the deterministic Algorithm 8.3.2, adapted to the max-cut problem, the parameter vectors $\mathbf{p}_0, \mathbf{p}_1, \ldots$ converge to the optimal $\mathbf{p}^* = (1,0,1,0,1)$ after two iterations, provided that $\varrho = 10^{-1}$ and $\mathbf{p}_0 = (1, 1/2, 1/2, 1/2, 1/2)$.

Iteration 1

In the first step of the first iteration, we have to determine γ_1 from

$$\gamma_t = \max \{\gamma \text{ s.t. } \mathbb{E}_{\mathbf{p}_{t-1}}[I_{\{S(\mathbf{X}) \geqslant \gamma\}}] \geqslant 0.1\} . \tag{8.31}$$

It is readily seen that under the parameter vector $\mathbf{p}_0$, $S(\mathbf{X})$ takes values in $\{0, 6, 9, 10, 11, 13, 14, 15, 16\}$ with probabilities $\{1/16, 3/16, 3/16, 1/16, 3/16, 1/16, 2/16, 1/16, 1/16\}$. Hence, we find $\gamma_1 = 15$. In the second step, we need to solve

$$\mathbf{p}_t = \underset{\mathbf{p}}{\operatorname{argmax}} \, \mathbb{E}_{\mathbf{p}_{t-1}} \left[I_{\{S(\mathbf{X}) \geqslant \gamma_t\}} \ln f(\mathbf{X}; \mathbf{p}) \right] , \tag{8.32}$$

which has the solution

$$p_{t,i} = \frac{\mathbb{E}_{\mathbf{p}_{t-1}}[I_{\{S(\mathbf{X}) \geqslant \gamma_t\}} X_i]}{\mathbb{E}_{\mathbf{p}_{t-1}}[I_{\{S(\mathbf{X}) \geqslant \gamma_t\}}]} .$$

There are only two vectors $\mathbf{x}$ for which $S(\mathbf{x}) \geqslant 15$, namely, $(1, 0, 0, 0, 1)$ and $(1, 0, 1, 0, 1)$, and both have probability 1/16 under $\mathbf{p}_0$. Thus,

$$p_{1,i} = \begin{cases} \dfrac{2/16}{2/16} = 1 & \text{for } i = 1, 5, \\ \dfrac{1/16}{2/16} = \dfrac{1}{2} & \text{for } i = 3, \\ \dfrac{0}{2/16} = 0 & \text{for } i = 4, 2. \end{cases}$$

Iteration 2

In the second iteration $S(\mathbf{X})$ is 15 or 16 with probability 1/2. Applying again (8.31) and (8.32) yields the optimal $\gamma_2 = 16$ and the optimal $\mathbf{p}_2 = (1, 0, 1, 0, 1)$, respectively.

Remark 8.4.1 (Alternative Stopping Rule) Note that the stopping rule (8.21), which is based on convergence of the sequence $\{\widehat{\gamma}_t\}$ to γ^*, stops Algorithm 8.3.1 when the sequence $\{\widehat{\gamma}_t\}$ does not change. An alternative stopping rule is to stop when the sequence $\{\widehat{\mathbf{p}}_t\}$ is very close to a degenerated one, for example if $\min\{\widehat{p}_i, 1 - \widehat{p}_i\} < \epsilon$ for all i, where ϵ is some small number.

The code in Table 8.10 gives a simple Matlab implementation of the CE algorithm for the max-cut problem, with cost matrix (8.27). It is important to note that, although the max-cut examples presented here are of relatively small size, basically the *same* CE program can be used to tackle max-cut problems of much higher dimension, comprising hundreds or thousands of nodes.

Table 8.10 Matlab CE program to solve the max-cut problem with cost matrix (8.27).

```
global C;
C = [ 0  2  2  5  0;                          % cost matrix
      2  0  1  0  3;
      2  1  0  4  2;
      5  0  4  0  1;
      0  3  2  1  0];
m = 5; N = 100; Ne = 10; eps = 10^-3; p = 1/2*ones(1,m); p(1) = 1;
while max(min(p,1-p)) > eps
   x = (rand(N,m) < ones(N,1)*p);             % generate cut vectors
   SX = S(x);
   sortSX = sortrows([x SX], m+1);
   p = mean(sortSX(N-Ne+1:N, 1:m))            % update the parameters
end

function perf = S(x)                          % performance function
global C;
N = size(x,1);
for i=1:N
   V1 = find(x(i,:));                         % {V1,V2} is the partition
   V2 = find(~x(i,:));
perf(i,1) = sum(sum(C(V1,V2)));               % size of the cut
end
```

■ EXAMPLE 8.9 Maximal Cuts for the Dodecahedron Graph

To further illustrate the behavior of the CE algorithm for the max-cut problem, consider the so-called *dodecahedron graph* in Figure 8.5. Suppose that all edges have cost 1. We wish to partition the node set into two subsets (color the nodes black and white) such that the cost across the cut, given by (8.28), is maximized. Although this problem exhibits a lot of symmetry, it is not clear beforehand what the solution(s) should be.

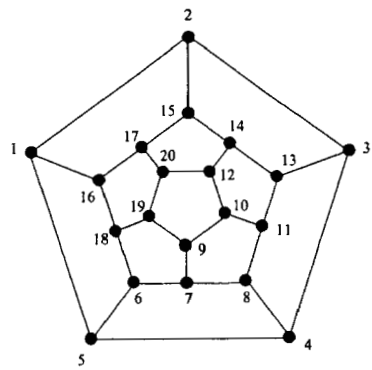


Figure 8.5 The dodecahedron graph.

The performance of the CE algorithm is depicted in Figure 8.6 using $N = 200$ and $\varrho = 0.1$.

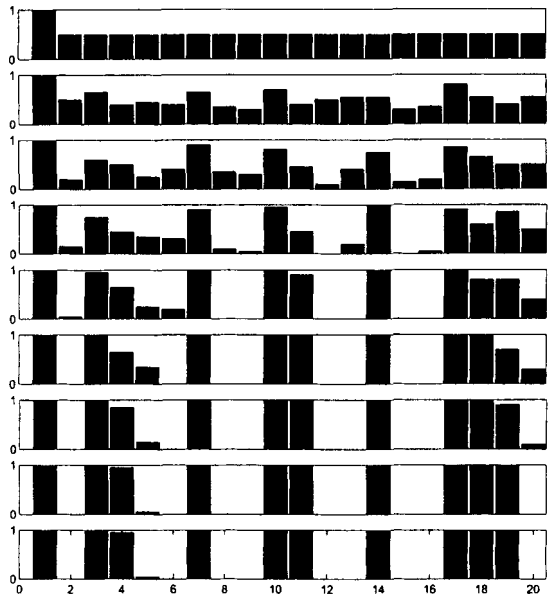


Figure 8.6 The evolution of the CE algorithm for the dodecahedron max-cut problem.

Observe that the probability vector $\widehat{\mathbf{p}}_t$ quickly (eight iterations) converges to a degenerate vector—corresponding (for this particular case) to the solution $\mathbf{x}^* = (1,0,1,1,0,0,1,0,0,1,1,0,\ 0,1,0,0,1,1,1,0)$. Thus, $V_1^* = \{1,3,4,7,10,11,14,17,18,19\}$. This required around 1600 function evaluations, as compared to $2^{19} - 1 \approx 5 \cdot 10^5$ if all cut vectors were to be enumerated. The maximal value is 24. It is interesting to note that, because of the symmetry, there are in fact many optimal solutions. We found that during each run the CE algorithm "focuses" on one (not always the same) of the solutions.

The Max-cut Problem with r Partitions

We can readily extend the max-cut procedure to the case where the node set V is partitioned into $r > 2$ subsets $\{V_1, \ldots, V_r\}$ such that the sum of the total weights of all edges going from subset V_a to subset V_b, $a, b = 1, \ldots, r$, $(a < b)$ is maximized. Thus, for each partition $\{V_1, \ldots, V_r\}$, the value of the objective function is

$$\sum_{a=1}^{r} \sum_{b=a+1}^{r} \sum_{i \in V_a,\, j \in V_b} c_{ij} \,.$$

In this case, one can follow the basic steps of Algorithm 8.3.1 using independent r-point distributions, instead of independent Bernoulli distributions, and update the probabilities as

$$\widehat{p}_{t,ij} = \frac{\sum_{\mathbf{X}_k \in \mathscr{E}_t} I_{\{X_{ki}=j\}}}{|\mathscr{E}_t|} \,. \tag{8.33}$$

8.5 THE PARTITION PROBLEM

The partition problem is similar to the max-cut problem. The only difference is that the *size* of each class is *fixed* in advance. This has implications for the trajectory generation. Consider, for example, a partition problem in which V has to be partitioned into two *equal* sets, assuming n is even. We could simply use Algorithm 8.4.1 for the random cut generation, that is, generate $\mathbf{X} \sim \mathsf{Ber}(\mathbf{p})$ and *reject* partitions that have unequal size, but this would be highly inefficient. We can speed up this method by drawing directly from the *conditional* distribution of $\mathbf{X} \sim \mathsf{Ber}(\mathbf{p})$ given $X_1 + \cdots + X_n = n/2$. The parameter $\mathbf{p}$ is then updated in exactly the same way as before. Unfortunately, generating from a conditional Bernoulli distribution is not as straightforward as generating independent Bernoulli random variables. A useful technique is the so-called *drafting* method. We provide computer code for this method in Section A.2 of the Appendix.

As an alternative, we describe next a simple algorithm for the generation of a random bipartition $\{V_1, V_2\}$ with exactly m elements in V_1 and $n - m$ elements in V_2 that works well in practice. Extension of the algorithm to r-partition generation is simple.

The algorithm requires the generation of random permutations $\Pi = (\Pi_1, \ldots, \Pi_n)$ of $(1, \ldots, n)$, uniformly over the space of all permutations. This can be done via Algorithm 2.8.2. We demonstrate our algorithm first for a five-node network, assuming $m = 2$ and $m - n = 3$ for a given vector $\mathbf{p} = (p_1, \ldots, p_5)$.

■ **EXAMPLE 8.10 Generating a Bi-Partition for $m = 2$ and $n = 5$**

1. Generate a random permutation $\Pi = (\Pi_1, \ldots, \Pi_5)$ of $(1, \ldots, 5)$, uniformly over the space of all 5! permutations. Let $(\pi_1, \ldots, \pi_5)$ be a particular outcome, for example, $(\pi_1, \ldots, \pi_5) = (3, 5, 1, 2, 4)$. This means that we shall draw independent Bernoulli random variables in the following order: $\mathsf{Ber}(p_3)$, $\mathsf{Ber}(p_5)$, $\mathsf{Ber}(p_1)$, ...

2. Given $\Pi = (\pi_1, \ldots, \pi_5)$ and the vector $\mathbf{p} = (p_1, \ldots, p_5)$, generate independent Bernoulli random variables $X_{\pi_1}, X_{\pi_2}, \ldots$ from $\mathsf{Ber}(p_{\pi_1})$, $\mathsf{Ber}(p_{\pi_2}), \ldots$, respectively, *until either exactly* $m = 2$ unities or $n - m = 3$ zeros are generated. Note that in general, the number of samples is a random variable with the range from $\min\{m, n - m\}$ to n. Assume for concreteness that the first four independent Bernoulli samples (from the above $\mathsf{Ber}(p_3)$, $\mathsf{Ber}(p_5)$, $\mathsf{Ber}(p_1)$, $\mathsf{Ber}(p_2)$) result in the following outcome $(0, 0, 1, 0)$. Since we have already generated three 0s, we can set $X_4 \equiv 1$ and deliver $\{V_1(\mathbf{X}), V_2(\mathbf{X})\} = \{(1, 4), (2, 3, 5)\}$ as the desired partition.

3. If in the previous step $m = 2$ unities are generated, set the remaining three elements to 0; if, on the other hand, three 0s are generated, set the remaining two elements to 1 and deliver $\mathbf{X} = (X_1, \ldots, X_n)$ as the final partition vector. Construct the partition $\{V_1(\mathbf{X}), V_2(\mathbf{X})\}$ of V.

With this example in hand, the random partition generation algorithm can be written as follows.

Algorithm 8.5.1 (Random Partition Generation Algorithm)

1. *Generate a random permutation* $\Pi = (\Pi_1, \ldots, \Pi_n)$ *of* $(1, \ldots, n)$ *uniformly over the space of all* $n!$ *permutations.*

2. *Given* $\Pi = (\pi_1, \ldots, \pi_n)$, *independently generate Bernoulli random variables* X_{π_1}, $X_{\pi_2}, \ldots$ *from* $\mathsf{Ber}(p_{\pi_1}), \mathsf{Ber}(p_{\pi_2}), \ldots$, *respectively,* until m *1s or* $n - m$ *0s are generated.*

3. *If in the previous step* m *1s are generated, set the remaining elements to 0; if, on the other hand,* $n - m$ *0s are generated, set the remaining elements to 1s. Deliver* $\mathbf{X} = (X_1, \ldots, X_n)$ *as the final partition vector.*

4. *Construct the partition* $\{V_1(\mathbf{X}), V_2(\mathbf{X})\}$ *of* V *and calculate the performance* $S(\mathbf{X})$ *according to* (8.28).

We take the updating formula for the reference vector $\mathbf{p}$ *exactly the same* as in (8.10).

8.5.1 Empirical Computational Complexity

Finally, let us discuss the computational complexity of Algorithm 8.3.1 for the max-cut and the partition problems, which can be defined as

$$\kappa_n = T_n(N_n G_n + U_n) \,. \tag{8.34}$$

Here T_n is the total number of iterations needed before Algorithm 8.3.1 stops; N_n is the sample size, that is, the total number of maximal cuts and partitions generated at each iteration; G_n is the cost of generating the random Bernoulli vectors of size n for Algorithm 8.3.1; $U_n = \mathcal{O}(N_n n^2)$ is the cost of updating the tuple $(\widehat{\gamma}_t, \widehat{\mathbf{p}}_t)$. The last follows from the fact that computing $S(\mathbf{X})$ in (8.28) is a $\mathcal{O}(n^2)$ operation.

For the model in (8.49) we found empirically that $T_n = \mathcal{O}(\ln n)$, provided that $100 \leqslant n \leqslant 1000$. For the max-cut problem, considering that we take $n \leqslant N_n \leqslant 10\,n$ and that G_n is $\mathcal{O}(n)$, we obtain $\kappa_n = \mathcal{O}(n^3 \ln n)$. In our experiments, the complexity we observed was more like

$$\kappa_n = \mathcal{O}(n \ln n) \,.$$

The partition problem has similar computational characteristics. It is important to note that these empirical complexity results are solely for the model with the cost matrix (8.49).

8.6 THE TRAVELING SALESMAN PROBLEM

The CE method can also be applied to solve the traveling salesman problem (TSP). Recall (see Example 6.12 for a more detailed formulation) that the objective is to find the shortest tour through all the nodes in a graph G. As in Example 6.12, we assume that the graph is complete and that each tour is represented as a permutation $\mathbf{x} = (x_1, \ldots, x_n)$ of $(1, \ldots, n)$. Without loss of generality we can set $x_1 = 1$, so that the set of all possible tours $\mathscr{X}$ has cardinality $|\mathscr{X}| = (n-1)!$. Let $S(\mathbf{x})$ be the total length of tour $\mathbf{x} \in \mathscr{X}$, and let $C = (c_{ij})$ be the cost matrix. Our goal is thus to solve

$$\min_{\mathbf{x} \in \mathscr{X}} S(\mathbf{x}) \quad = \quad \min_{\mathbf{x} \in \mathscr{X}} \left\{ \sum_{i=1}^{n-1} c_{x_i, x_{i+1}} + c_{x_n, 1} \right\} \,. \tag{8.35}$$

In order to apply the CE algorithm, we need to specify a parameterized random mechanism to generate the random tours. As mentioned, the updating formulas for the parameters follow, as always, from CE minimization.

An easy way to explain how the tours are generated and how the parameters are updated is to relate (8.35) to an *equivalent* minimization problem. Let

$$\widetilde{\mathscr{X}} = \{(x_1, \ldots, x_n) \,:\, x_1 = 1, \quad x_i \in \{1, \ldots, n\}\,, \quad i = 2, \ldots, n\} \tag{8.36}$$

be the set of vectors that correspond to tours that start in 1 and can visit the same city more than once. Note that $|\widetilde{\mathscr{X}}| = n^{n-1}$ and $\mathscr{X} \subset \widetilde{\mathscr{X}}$. When $n = 4$, we could have, for example, $\mathbf{x} = (1, 3, 1, 3) \in \widetilde{\mathscr{X}}$, corresponding to the *path* (not tour) $1 \to 3 \to 1 \to 3 \to 1$. Define the function $\widetilde{S}$ on $\widetilde{\mathscr{X}}$ by $\widetilde{S}(\mathbf{x}) = S(\mathbf{x})$, if $\mathbf{x} \in \mathscr{X}$ and $\widetilde{S}(\mathbf{x}) = \infty$ otherwise. Then, obviously, (8.35) is equivalent to the minimization problem

$$\text{minimize } \widetilde{S}(\mathbf{x}) \text{ over } \mathbf{x} \in \widetilde{\mathscr{X}}\,. \tag{8.37}$$

A simple method to generate a random path $\mathbf{X} = (X_1, \ldots, X_n)$ in $\widetilde{\mathscr{X}}$ is to use a Markov chain on the graph G, starting at node 1 and stopping after n steps. Let $\mathbf{P} = (p_{ij})$ denote the one-step transition matrix of this Markov chain. We assume that the diagonal elements of $\mathbf{P}$ are 0 and that all other elements of $\mathbf{P}$ are strictly positive, but otherwise $\mathbf{P}$ is a general $n \times n$ stochastic matrix.

The pdf $f(\cdot; \mathbf{P})$ of $\mathbf{X}$ is thus parameterized by the matrix $\mathbf{P}$, and its logarithm is given by

$$\ln f(\mathbf{x}; \mathbf{P}) = \sum_{r=1}^{n} \sum_{i,j} I_{\{\mathbf{x} \in \widetilde{\mathscr{X}}_{ij}(r)\}} \ln p_{ij}\,,$$

where $\widetilde{\mathscr{X}}_{ij}(r)$ is the set of all paths in $\widetilde{\mathscr{X}}$ for which the r-th transition is from node i to j. The updating rules for this modified optimization problem follow from (8.18), with $\{S(\mathbf{X}_i) \geqslant \gamma_t\}$ replaced with $\{\widetilde{S}(\mathbf{X}_i) \leqslant \gamma_t\}$, under the condition that the rows of $\mathbf{P}$ sum up to 1. Using Lagrange multipliers $u_1, \ldots, u_n$, we obtain the maximization problem

$$\max_{\mathbf{P}} \min_{u_1, \ldots, u_n} \left\{ \mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \ln f(\mathbf{X}; \mathbf{P}) \right] + \sum_{i=1}^{n} u_i \left(\sum_{j=1}^{n} p_{ij} - 1 \right) \right\}\,. \tag{8.38}$$

Differentiating the expression within braces above with respect to p_{ij} yields, for all $j = 1, \ldots, n$,

$$\frac{\mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X} \in \widetilde{\mathscr{X}}_{ij}(r)\}} \right]}{p_{ij}} + u_i = 0\,. \tag{8.39}$$

Summing over $j = 1, \ldots, n$ gives $\mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X} \in \widetilde{\mathscr{X}}_i(r)\}} \right] = -u_i$, where $\widetilde{\mathscr{X}}_i(r)$ is the set of paths for which the r-th transition starts from node i. It follows that the optimal p_{ij} is given by

$$p_{ij} = \frac{\mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X} \in \widetilde{\mathscr{X}}_{ij}(r)\}} \right]}{\mathbb{E}_{\mathbf{P}} \left[I_{\{\widetilde{S}(\mathbf{X}) \leqslant \gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X} \in \widetilde{\mathscr{X}}_i(r)\}} \right]}\,. \tag{8.40}$$

The corresponding estimator is

$$\widehat{p}_{ij} = \frac{\sum_{k=1}^{N} I_{\{\widetilde{S}(\mathbf{X}_k)\leqslant\gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X}_k\in\widetilde{\mathscr{X}}_{ij}(r)\}}}{\sum_{k=1}^{N} I_{\{\widetilde{S}(\mathbf{X}_k)\leqslant\gamma\}} \sum_{r=1}^{n} I_{\{\mathbf{X}_k\in\widetilde{\mathscr{X}}_{i}(r)\}}} . \tag{8.41}$$

This has a very simple interpretation. To update p_{ij}, we simply take the fraction of times in which the transition from i to j occurs, taking into account only those paths that have a total length less than or equal to γ.

This is how one could, *in principle*, carry out the sample generation and parameter updating for problem (8.37): generate paths via a Markov process with transition matrix $\mathbf{P}$ and use the updating formula (8.41). However, *in practice*, we would never generate the tours this way, since most paths would visit cities (other than 1) more than once, and therefore their $\widetilde{S}$ values would be ∞ — that is, most of the paths would not constitute tours. In order to avoid the generation of irrelevant paths, we proceed as follows.

Algorithm 8.6.1 (Trajectory Generation Using Node Transitions)

1. *Define* $\mathbf{P}^{(1)} = \mathbf{P}$ *and* $X_1 = 1$. *Let* $k = 1$.

2. *Obtain* $\mathbf{P}^{(k+1)}$ *from* $\mathbf{P}^{(k)}$ *by first setting the* X_k*-th column of* $\mathbf{P}^{(k)}$ *to 0 and then normalizing the rows to sum up to 1. Generate* X_{k+1} *from the distribution formed by the* X_k*-th row of* $\mathbf{P}^{(k)}$.

3. *If* $k = n - 1$, *then* **stop**; *otherwise, set* $k = k + 1$ *and reiterate from Step 2.*

A fast implementation of the above algorithm, due to Radislav Vaisman, is given by the following procedure, which has complexity $\mathcal{O}(n^2)$. Here i is the currently visited node, and $(b_1, \ldots, b_n)$ is used to keep track of which states have been visited: $b_i = 1$ if node i has already been visited and 0 otherwise.

Procedure (Fast Generation of Trajectories)

```
Let t = 1, b_1 = 1, b_j = 0, for all j ≠ 1, i = 1, and X_1 = 1
Generate U ~ U(0,1), and let R = U Σ_{j=1}^{n} (1 − b_j) p_ij
Let sum = 0 and j = 0
while sum < R do
    j = j + 1
    if b_j = 0
        sum = sum + p_ij
    end
end
Set t = t + 1, X_t = j, b_j = 1 and i = j
if t = n
    stop
else return to 2
end
```

It is important to realize that the updating formula for p_{ij} remains the same. By using Algorithm 8.6.1, we are merely *speeding up* our naive trajectory generation by only generating *tours*. As a consequence, each trajectory will visit each city once, and transitions from i to j can at most occur once. It follows that

$$\widetilde{\mathscr{X}}_{ij}(r) = \widetilde{\mathscr{X}}_{i}(r) = \emptyset, \quad \text{for } r \geqslant 2 ,$$

so that the updating formula for p_{ij} can be written as

$$\widehat{p}_{ij} = \frac{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \leqslant \gamma\}} I_{\{\mathbf{X}_k \in \mathscr{X}_{ij}\}}}{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \leqslant \gamma\}}} , \tag{8.42}$$

where $\mathscr{X}_{ij}$ is the set of tours in which the transition from i to j is made. This has the same "natural" interpretation discussed for (8.41).

For the initial matrix $\widehat{\mathbf{P}}_0$, one could simply take all off-diagonal elements equal to $1/(n-1)$, provided that all cities are connected.

Note that ϱ and α should be chosen as in Remark 8.3.3, and the sample size for TSP should be $N = c\,n^2$, with $c > 1$, say $c = 5$.

■ **EXAMPLE 8.11 TSP on Hammersley Points**

To shed further light on the CE method applied to the TSP, consider a shortest (in Euclidean distance sense) tour through a set of *Hammersley points*. These form an example of *low-discrepancy* sequences that cover a d-dimensional unit cube in a pseudo-random but orderly way. To find the 2^5 two-dimensional Hammersley points of order 5, construct first the x-coordinates by taking all binary fractions $x = 0.x_1x_2 \ldots x_5$. Then let the corresponding y coordinate be obtained from x by reversing the binary digits. For example, if $x = 0.11000$ (binary), which is $x = 1/2 + 1/4 = 3/4$ (decimal), then $y = 0.00011$ (binary), which is $y = 3/32$ (decimal). The Hammersley points, in order of increasing y, are thus

$$\{(0,0),(16,1),(8,2),(24,3),(4,4),(20,5),(12,6),(28,7),(2,8),(18,9),(10,10),$$
$$(26,11),(6,12),(22,13),(14,14),(30,15),(1,16),(17,17),(9,18),(25,19),(5,20),$$
$$(21,21),(13,22),(29,23),(3,24),(19,25),(11,26),(27,27),(7,28),(23,29),$$
$$(15,30),(31,31)\}/32 .$$

Table 8.11 and Figure 8.7 show the behavior of the CE algorithm applied to the Hammersley TSP. In particular, Table 8.11 depicts the progression of $\widehat{\gamma}_t$ and S_t^b, which denote the largest of the elite values in iteration t and the best value encountered so far, respectively. Similarly, Figure 8.7 shows the evolution of the transition matrices $\mathbf{P}_t$. Here the initial elements $p_{0,ij}$, $i \neq j$ are all set to $1/(n-1) = 1/31$; the diagonal elements are 0. We used a sample size of $N = 5\,n^2 = 5120$, rarity parameter $\varrho = 0.03$, and smoothing parameter $\alpha = 0.7$. The algorithm was stopped when no improvement in $\widehat{\gamma}_t$ during three consecutive iterations was observed.

Table 8.11 Progression of the CE algorithm for the Hammersley TSP.

t	S_t^b	$\widehat{\gamma}_t$	t	S_t^b	$\widehat{\gamma}_t$
1	11.0996	13.2284	16	5.95643	6.43456
2	10.0336	11.8518	17	5.89489	6.31772
3	9.2346	10.7385	18	5.83683	6.22153
4	8.27044	9.89423	19	5.78224	6.18498
5	7.93992	9.18102	20	5.78224	6.1044
6	7.54475	8.70609	21	5.78224	6.0983
7	7.32622	8.27284	22	5.78224	6.06036
8	6.63646	7.94316	23	5.78224	6.00794
9	6.63646	7.71491	24	5.78224	5.91265
10	6.61916	7.48252	25	5.78224	5.86394
11	6.43016	7.25513	26	5.78224	5.86394
12	6.20255	7.07624	27	5.78224	5.83645
13	6.14147	6.95727	28	5.78224	5.83645
14	6.12181	6.76876	29	5.78224	5.83645
15	6.02328	6.58972			

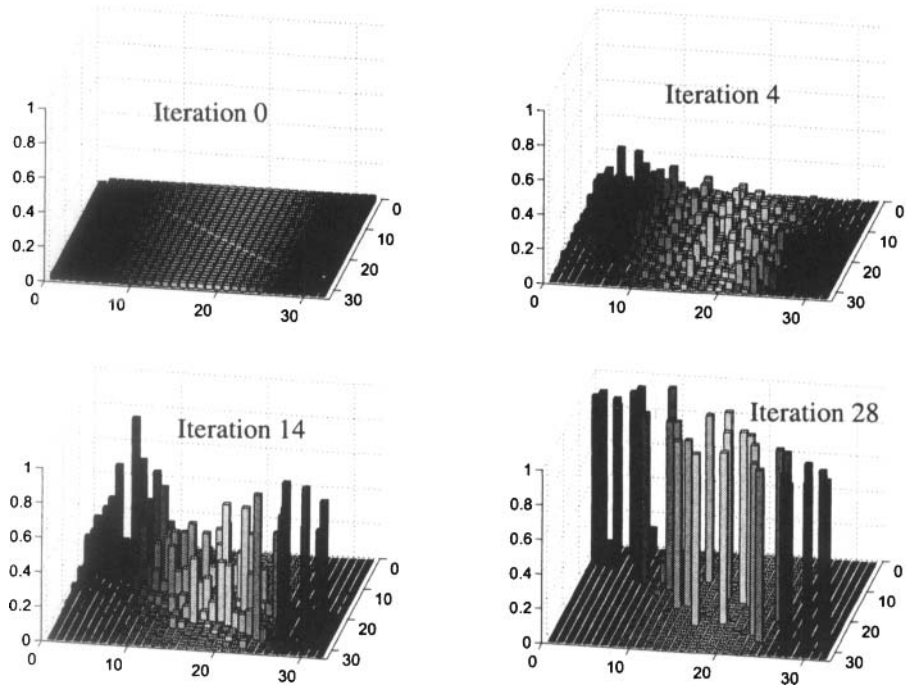


Figure 8.7 Evolution of $\mathbf{P}_t$ in the CE algorithm for the Hammersley TSP.

The optimal tour length for the Hammersley problem is $\gamma^* = 5.78224$ (rounded), which coincides with $\widehat{\gamma}_{29}$ found in Table 8.11. A corresponding solution (optimal tour) is (1, 5, 9, 17, 13, 11, 15, 18, 22, 26, 23, 19, 21, 25, 29, 27, 31, 30, 32, 28, 24, 20, 16, 8, 12, 14, 10, 6, 4, 2, 7, 3), depicted in Figure 8.8. There are several other optimal tours (see Problem 8.13) but all exhibit a straight line through the points (10,10)/32, (14,14)/32, (17,17)/32 and (21,21)/32.

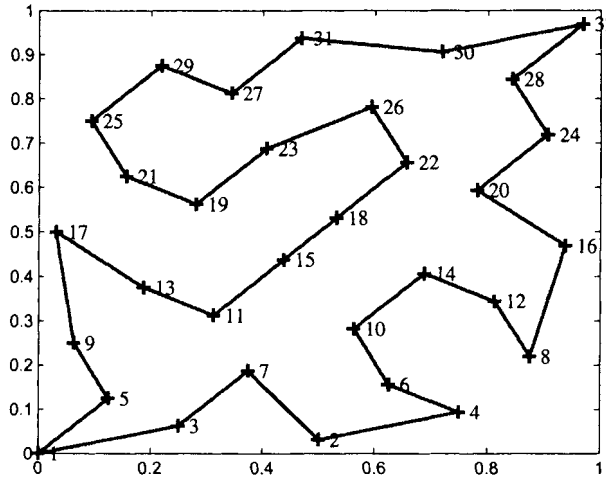


Figure 8.8 An optimal tour through the Hammersley points.

8.6.1 Incomplete Graphs

The easiest way to deal with TSPs on incomplete graphs is, as already remarked in Example 6.12, to make the graph complete by acquiring extra links with infinite cost. However, if many entries in the cost matrix are infinite, most of the generated tours in Algorithm 8.6.1 will initially be invalid (yield a length of ∞). A better way of choosing $\mathbf{P}_0 = (p_{0,ij})$ is then to assign smaller initial probabilities to pairs of nodes for which no direct link exists. In particular, let d_i be the *degree* of node i, that is, the number of finite entries in the i-th row of the matrix C. We can then proceed as follows:

1. If $c_{ij} = \infty$, set $p_{0,ij}$ to $\frac{1-\delta}{d_i}$, where δ is a small number, say $\delta = 0.1$. Set the remaining elements to ε, except for $p_{0,ii} = 0$. Since the rows of $\mathbf{P}_0$ sum up to 1, we have $\varepsilon = \frac{\delta}{n-d_i-1}$.

2. Keep the above $p_{0,ij} = \varepsilon = \frac{\delta}{n-d_i-1}$ for *all iterations* of the CE Algorithm 8.3.1.

Since δ is the sum of all $p_{t,ij}$ corresponding to the ∞ elements in the i-th row of C, and since all such $p_{t,ij}$ are equal to each other (ε), we can generate a transition from each state i using only a $(d_i + 1)$-point distribution rather than the n-point distribution formed by the i-th row of $\widehat{\mathbf{P}}_t$. Indeed, if we *relabel* the elements of this row such that the first d_i entries

correspond to existing links, while the next $n - d_i - 1$ correspond to nonexisting links, then we obtain the following faster procedure for generating transitions.

Algorithm 8.6.2 (A Fast Procedure for Generating Transitions)

1. Generate a random variable $U \sim \mathsf{U}(0,1)$.
2. If $U \leqslant 1 - \delta$, generate the next transition from the discrete d_i point pdf with probabilities $p_{t,ij}/(1-\delta)$, $j = 1, \ldots, d_i$.
3. If $U > 1 - \delta$, generate the next transition by drawing a discrete random variable Z uniformly distributed over the points $d_i + 1, \ldots, n-1$ (recall that these points correspond to the ∞ elements in the i-th row of C).

It is important to note that the small elements of $\mathbf{P}_0$ corresponding to infinities in matrix C should be kept the same from iteration to iteration rather than being updated. By doing so, one obtains considerable speedup in trajectory generation.

8.6.2 Node Placement

We now present an alternative algorithm for trajectory generation due to Margolin [20] called the node *placement algorithm*. In contrast to Algorithm 8.6.1, which generates *transitions* from node to node (based on the transition matrix $P = (p_{ij})$), in Algorithm 8.6.3 below, a similar matrix

$$\mathbf{P} = \begin{pmatrix} p_{(1,1)} & p_{(1,2)} & \cdots & p_{(1,n)} \\ p_{(2,1)} & p_{(2,2)} & \cdots & p_{(2,n)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{(n,1)} & p_{(n,2)} & \cdots & p_{(n,n)} \end{pmatrix} \tag{8.43}$$

generates *node placements*. Specifically, $p_{(i,j)}$ corresponds to the probability of node i being visited at the j-th place in a tour of n cities. In other words, $p_{(i,j)}$ can be viewed as the probability that city (node) i is "arranged" to be visited at the j-th place in a tour of n cities. More formally, a *node placement vector* is a vector $\mathbf{y} = (y_1, \ldots, y_n)$ such that y_i denotes the place of node i in the tour $\mathbf{x} = (x_1, \ldots, x_n)$. The precise meaning is given by the correspondence

$$y_i = j \iff x_j = i\,, \tag{8.44}$$

for all $i, j \in \{1, \ldots, n\}$. For example, the node placement vector $\mathbf{y} = (3,4,2,6,5,1)$ in a six-node network defines uniquely the tour $\mathbf{x} = (6,3,1,2,5,4)$. The performance of each node placement $\mathbf{y}$ can be defined as $\overline{S}(\mathbf{y}) = S(\mathbf{x})$, where $\mathbf{x}$ is the unique path corresponding to $\mathbf{y}$.

Algorithm 8.6.3 (Trajectory Generation Using Node Placements)

1. *Define* $\mathbf{P}^{(1)} = \mathbf{P}$. *Let* $k = 1$.
2. *Generate* Y_k *from the distribution formed by the k-th row of* $\mathbf{P}^{(k)}$. *Obtain the matrix* $\mathbf{P}^{(k+1)}$ *from* $\mathbf{P}^{(k)}$ *by first setting the* Y_k*-th column of* $\mathbf{P}^{(k)}$ *to 0 and then normalizing the rows to sum up to 1.*
3. *If* $k = n$ *then* **stop**; *otherwise, set* $k = k + 1$ *and reiterate from Step 2.*
4. *Determine the tour by* (8.44) *and evaluate the length of the tour by* (8.35).

It is readily seen that the updating formula for $p_{(i,j)}$ is now

$$\widehat{p}_{(i,j)} = \frac{\sum_{k=1}^{N} I_{\{\overline{S}(\mathbf{Y}_k) \leqslant \gamma\}} I_{\{Y_{ki}=j\}}}{\sum_{k=1}^{N} I_{\{\overline{S}(\mathbf{Y}_k) \leqslant \gamma\}}} . \tag{8.45}$$

Our simulation results with the TSP and other problems do not indicate clear superiority of either Algorithm 8.6.1 or Algorithm 8.6.3 in terms of the efficiency (speed and accuracy) of the main CE Algorithm 8.3.1.

8.6.3 Case Studies

To illustrate the accuracy and robustness of the CE algorithm, we applied the algorithm to a number of benchmark problems from the TSP library

`http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/`

In all cases the *same* set of CE parameters were chosen: $\varrho = 0.03$, $\alpha = 0.7$, $N = 5\,n^2$, and we use the stopping rule (8.21) with the parameter $d = 3$.

Table 8.12 presents the performance of Algorithm 8.3.1 for a selection of *symmetric* TSPs from this library. To study the variability in the solutions, each problem was repeated 10 times. In the table, *min*, *mean* and *max* denote the smallest (that is, best), average, and largest of the 10 estimates for the optimal value. The true optimal value is denoted by γ^*.

The average CPU time in seconds and the average number of iterations are given in the last two columns. The size of the problem (number of nodes) is indicated in its name. For example, `st70` has $n = 70$ nodes. Similar case studies for the *asymmetric case* may be found in Table 2.5 of [31].

Table 8.12 Case studies for the TSP.

file	γ^*	min	mean	max	CPU	$\bar{T}$
burma14	3323	3323	3325.6	3336	0.14	12.4
ulysses16	6859	6859	6864	6870	0.21	14.1
ulysses22	7013	7013	7028.9	7069	1.18	22.1
bayg29	1610	1610	1628.6	1648	4.00	28.2
bays29	2020	2020	2030.9	2045	3.83	27.1
dantzig42	699	706	717.7	736	19.25	38.4
eil51	426	428	433.9	437	65.0	63.35
berlin52	7542	7618	7794	8169	64.55	59.9
st70	675	716	744.1	765	267.5	83.7
eil76	538	540	543.5	547	467.3	109.0
pr76	108159	109882	112791	117017	375.3	88.9

At this end, note that CE is ideally suitable for parallel computation, since parallel computing speeds up the process by almost a factor of r, where r is the number of parallel processors.

One might wonder why the CE Algorithm 8.2.1, with such simple updating rules and quite arbitrary parameters α and ϱ, performs so nicely for combinatorial optimization problems. A possible explanation is that the objective function S for combinatorial optimization problems is typically close to being additive; see, for example, the objective function S for the TSP problem in (8.35). For other optimization problems (for example, optimizing complex multiextremal continuous functions), one needs to make a more careful and more conservative choice of the parameters α and ϱ.

8.7 CONTINUOUS OPTIMIZATION

We will briefly discuss how the CE method can be applied to solve continuous optimization problems. Let $S(\mathbf{x})$ be a real-valued function on $\mathbb{R}^n$. To maximize the function via CE, one must specify a family of parameterized distributions to generate samples in $\mathbb{R}^n$. This family must include, at least in the limiting case, the degenerate distribution that puts all its probability mass on an optimal solution. A simple choice is to use a multivariate normal distribution, parameterized by a mean vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ and a covariance matrix Σ. When the covariance matrix is chosen to be diagonal — that is, the components of $\mathbf{X}$ are independent — the CE updating formulas become particularly easy. In particular, denoting $\{\mu_i\}$ and $\{\sigma_i\}$ the means and standard deviations of the components, the updating formulas are (see Problem 8.17)

$$\widehat{\mu}_{t,i} = \frac{\sum_{\mathbf{X}_k \in \mathscr{E}_t} X_{ki}}{|\mathscr{E}_t|} \qquad i = 1, \ldots, n\,, \tag{8.46}$$

and

$$\widehat{\sigma}_{t,i} = \sqrt{\frac{\sum_{\mathbf{X}_k \in \mathscr{E}_t} (X_{ki} - \widehat{\mu}_{t,i})^2}{|\mathscr{E}_t|}}, \qquad i = 1, \ldots, n\,, \tag{8.47}$$

where X_{ki} is the i-th component of $\mathbf{X}_k$ and $\mathbf{X}_1, \ldots, \mathbf{X}_n$ is a random sample from $\mathsf{N}(\widehat{\boldsymbol{\mu}}_{t-1}, \widehat{\Sigma}_{t-1})$. In other words, the means and standard deviations are simply updated via the corresponding maximum likelihood estimators based on the elite samples $\mathscr{E}_t = \{\mathbf{X}_k : S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}$.

■ EXAMPLE 8.12 The Peaks Function

Matlab's peaks function,

$$\begin{aligned} S(\mathbf{x}) = {} & 3\,(1 - x_1)^2 \exp(-(x_1^2) - (x_2 + 1)^2) - 10\,(x_1/5 - x_1^3 - x_2^5) \exp(-x_1^2 - x_2^2) \\ & - 1/3 \exp(-(x_1 + 1)^2 - x_2^2)\,, \end{aligned}$$

has various local maxima. In Section A.5 of the Appendix, a simple Matlab implementation of CE Algorithm 8.3.1 is given for finding the global maximum of this function, which is approximately $\gamma^* = 8.10621359$ and is attained at $\mathbf{x}^* = (-0.0093151, 1.581363)$. The choice of the initial value for $\boldsymbol{\mu}$ is not important, but the initial standard deviations should be chosen large enough to ensure initially a close to uniform sampling of the region of interest. The CE algorithm is

stopped when all standard deviations of the sampling distribution are less than some small ε.

Figure 8.9 gives the evolution of the worst and best of the elite samples, that is, $\widehat{\gamma}_t$ and S_t^*, for each iteration t. We see that the values quickly converge to the optimal value γ^*.

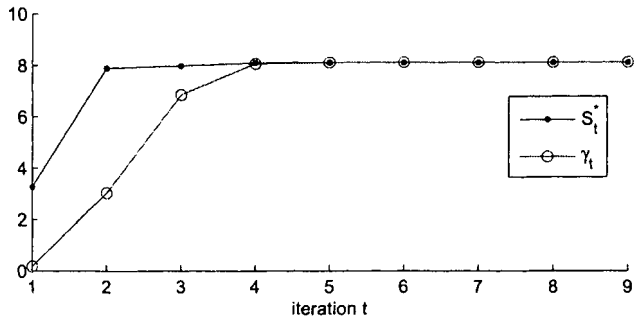


Figure 8.9 Evolution of the CE algorithm for the peaks function.

Remark 8.7.1 (Injection) When using the CE method to solve practical optimization problems with many constraints and many local optima, it is sometimes necessary to prevent the sampling distribution from shrinking too quickly. A simple but effective approach is the following *injection* method [3]. Let S_t^* denote the best performance found at the t-th iteration, and (in the normal case) let σ_t^* denote the largest standard deviation at the t-th iteration. If σ_t^* is sufficiently small and $|S_t^* - S_{t-1}^*|$ is also small, then add some small value to each standard deviation, for example a constant δ or the value $c\,|S_t^* - S_{t-1}^*|$, for some fixed δ and c. When using CE with injection, a possible stopping criterion is to stop after a fixed number of injections.

8.8 NOISY OPTIMIZATION

One of the distinguishing features of the CE Algorithm 8.3.1 is that it can easily handle noisy optimization problems, that is, when the objective function $S(\mathbf{x})$ is corrupted with noise. We denote such a noisy function by $\widehat{S}(\mathbf{x})$. We assume that for each $\mathbf{x}$ we can readily obtain an outcome of $\widehat{S}(\mathbf{x})$, for example via generation of some additional random vector $\mathbf{Y}$, whose distribution may depend on $\mathbf{x}$.

A classical example of noisy optimization is simulation-based optimization [32]. A typical instance is the *buffer allocation problem*, where the objective is to allocate n buffer spaces among the $m-1$ "niches" (storage areas) between m machines in a serial production line so as to optimize some performance measure, such as the steady-state throughput. This performance measure is typically not available analytically and thus must be estimated via simulation. A detailed description of the buffer allocation problem, and of how CE can be used to solve this problem, is given in [31].

Another example is the noisy TSP, where, say, the cost matrix (c_{ij}), denoted now by $\mathbf{Y} = (Y_{ij})$, is random. Think of Y_{ij} as the random time to travel from city i to city j. The

total cost of a tour $\mathbf{x} = (x_1, \ldots, x_n)$ is given by

$$\widehat{S}(\mathbf{x}) = \sum_{i=1}^{n-1} Y_{x_i, x_{i+1}} + Y_{x_n, x_1} \,. \tag{8.48}$$

We assume that $\mathbb{E}[Y_{ij}] = c_{ij}$.

The main CE optimization Algorithm 8.3.1 for deterministic functions $S(\mathbf{x})$ is also valid for noisy ones $\widehat{S}(\mathbf{x})$. Extensive numerical studies [31] with the noisy version of Algorithm 8.3.1 show that it works nicely, because during the course of the optimization it *filters out* efficiently the noise component from $\widehat{S}(\mathbf{x})$. However, to get reliable estimates of the optimal solution of combinatorial optimization problems, one is required to increase the sample size N by a factor 2 to 5 in each iteration of Algorithm 8.3.1. Clearly, this factor increases with the "power" of the noise.

■ EXAMPLE 8.13 Noisy TSP

Suppose that in the first test case of Table 8.12, `burma14`, some uniform noise is added to the cost matrix. In particular, suppose that the cost of traveling from i to j is given by $Y_{ij} \sim \mathrm{U}(c_{ij} - 8, c_{ij} + 8)$, where c_{ij} is the cost for the deterministic case. The expected cost is thus $\mathbb{E}[Y_{ij}] = c_{ij}$, and the total cost $\widehat{S}(\mathbf{x})$ of a tour $\mathbf{x}$ is given by (8.48). The CE algorithm for optimizing the unknown $S(\mathbf{x}) = \mathbb{E}[\widehat{S}(\mathbf{x})]$ remains exactly the same as in the deterministic case, except that $S(\mathbf{x})$ is replaced with $\widehat{S}(\mathbf{x})$ and a different stopping criterion than (8.21) needs to be employed. A simple rule is to stop when the transition probabilities $\widehat{p}_{t,ij}$ satisfy $\min(\widehat{p}_{t,ij}, 1 - \widehat{p}_{t,ij}) < \varepsilon$ for *all* i and j, similar to Remark 8.4.1. We repeated the experiment 10 times, taking a sample size twice as large as for the deterministic case, that is, $N = 10 \cdot n^2$. For the above stopping criterion we took $\varepsilon = 0.02$. The other parameters remained the same as those described in Section 8.6.3. CE found the optimal solution eight times, which is comparable to its performance in the deterministic case.

Figure 8.10 displays the evolution of the worst performance of the elite samples $(\widehat{\gamma}_t)$ for both the deterministic and noisy case denoted by $\widehat{\gamma}_{1t}$ and $\widehat{\gamma}_{2t}$, respectively.

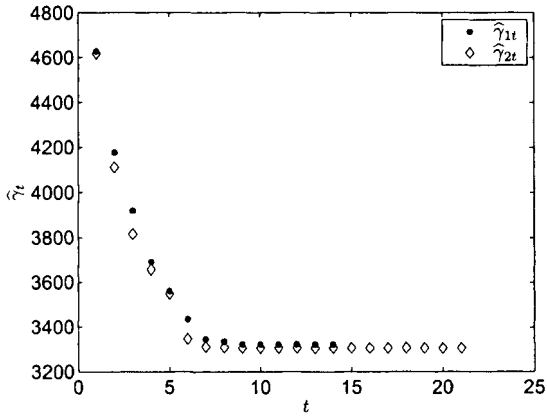


Figure 8.10 Evolution of the worst of the elite samples for a deterministic and noisy TSP.

We see in both cases a similar rapid drop in the level $\widehat{\gamma}_t$. It is important to note, however, that even though here the algorithm in both the deterministic and noisy cases converges to the optimal solution, the $\{\widehat{\gamma}_{2t}\}$ for the noisy case do not converge to $\gamma^* = 3323$, in contrast to the $\{\widehat{\gamma}_{1t}\}$ for the deterministic case. This is because the latter estimates eventually the $(1-\varrho)$-quantile of the deterministic $S(\mathbf{x}^*)$, whereas the former estimates the $(1-\varrho)$-quantile of $\widehat{S}(\mathbf{x}^*)$, which is random. To estimate $S(\mathbf{x}^*)$ in the noisy case, one needs to take the sample average of $\widehat{S}(\mathbf{x}_T)$, where $\mathbf{x}_T$ is the solution found at the final iteration.

PROBLEMS

8.1 In Example 8.2, show that the true CE-optimal parameter for estimating $\mathbb{P}(X \geqslant 32)$ is given by $v^* = 33$.

8.2 Write a CE program to reproduce Table 8.1 in Example 8.2. Use the final reference parameter $\widehat{v}_3$ to estimate ℓ via importance sampling, using a sample size of $N_1 = 10^6$. Estimate the relative error and give an approximate 95% confidence interval. Check if the true value of ℓ is contained in this interval.

8.3 In Example 8.2 calculate the exact relative error for the importance sampling estimator $\widehat{\ell}$ when using the CE optimal parameter $v^* = 33$ and compare it with the one estimated in Problem 8.2. How many samples are required to estimate ℓ with the same relative error, using CMC?

8.4 Implement the CE Algorithm 8.2.1 for the stochastic shortest path problem in Example 8.5 and reproduce Table 8.3.

8.5 Slightly modify the program used in Problem 8.4 to allow Weibull-distributed lengths. Reproduce Table 8.4 and make a new table for $\alpha = 5$ and $\gamma = 2$ (the other parameters remain the same).

8.6 Make a table similar to Table 8.4 by employing the standard CE method. That is, take $\text{Weib}(\alpha, v_i^{-1})$ as the importance sampling distribution for the i-th component and update the $\{v_i\}$ via (8.6).

8.7 Consider again the stochastic shortest path problem in Example 8.5, but now with nominal parameter $\mathbf{u} = (0.25, 0.5, 0.1, 0.3, 0.2)$. Implement the root-finding Algorithm 8.2.3 to estimate for which level γ the probability ℓ is equal to 10^{-5}. Also, give a 95% confidence interval for γ, for example, using the bootstrap method.

8.8 Adapt the cost matrix in the max-cut program of Table 8.10 and apply it to the dodecahedron max-cut problem in Example 8.9. Produce various optimal solutions and find out how many of these exist in total, disregarding the fivefold symmetry.

8.9 Consider the following symmetric cost matrix for the max-cut problem:

$$C = \begin{pmatrix} Z_{11} & B_{12} \\ B_{21} & Z_{22} \end{pmatrix}, \tag{8.49}$$

where Z_{11} is an $m \times m$ ($m < n$) symmetric matrix in which all the upper-diagonal elements are generated from a $\mathsf{U}(a, b)$ distribution (and all the lower-diagonal elements follow by symmetry), Z_{22} is an $(n-m) \times (n-m)$ symmetric matrix that is generated in the same way as Z_{11}, and all the other elements are c, apart from the diagonal elements, which are 0.

a) Show that if $c > b(n-m)/m$, the optimal cut is given by $V^* = \{\{1,\ldots,m\},\{m+1,\ldots,n\}\}$.

b) Show that the optimal value of the cut is $\gamma^* = c\,m\,(n-m)$.

c) Implement and run the CE algorithm on this synthetic max-cut problem for a network with $n = 400$ nodes, with $m = 200$. Generate Z_{11} and Z_{22} from the $\mathsf{U}(0,1)$ distribution and take $c = 1$. For the CE parameters take $N = 1000$ and $\varrho = 0.1$. List for each iteration the best and worst of the elite samples and the Euclidean distance $\|\widehat{\mathbf{p}}_t - \mathbf{p}^*\| = \sqrt{(\widehat{p}_{t,i} - p_i^*)^2}$ as a measure of how close the reference vector is to the optimal reference vector $\mathbf{p}^* = (1,1,\ldots,1,0,0,\ldots,0)$.

8.10 Consider a TSP with cost matrix $C = (c_{ij})$ defined by $c_{i,i+1} = 1$ for all $i = 1,2,\ldots,n-1$, and $c_{n,1} = 1$, while the remaining elements $c_{ij} \sim \mathsf{U}(a,b)$, $j \neq i+1$, $1 < a < b$, and $c_{ii} \equiv 0$.

a) Verify that the optimal permutation/tour is given by $\mathbf{x}^* = (1,2,3,\ldots,n)$, with minimal value $\gamma^* = n$.

b) Implement a CE algorithm to solve an instance of this TSP for the case $n = 30$ and make a table of the performance, listing the best and worst of the elite samples at each iteration, as well as

$$p_t^{mm} = \min_{1\leqslant i\leqslant n} \max_{1\leqslant j\leqslant n} \widehat{p}_{t,ij}\ ,$$

$t = 1,2,\ldots$, which corresponds to the min max value of the elements of the matrix $\widehat{\mathbf{P}}_t$ at iteration t. Use $d = 3$, $\varrho = 0.01$, $N = 4500$, and $\alpha = 0.7$. Also, keep track of the overall best solution.

8.11 Run Algorithm 8.3.1 on the data from the URL

`http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/atsp/`

and obtain a table similar to Table 8.12.

8.12 Select a TSP of your choice. Verify the following statements about the choice of CE parameters:

a) By reducing ϱ or increasing α, the convergence is faster but we can be trapped in a local minimum.

b) By reducing ϱ, one needs to decrease simultaneously α, and vice versa, in order to avoid convergence to a local minimum.

c) By increasing the sample size N, one can simultaneously reduce ϱ or (and) increase α.

8.13 Find out how many optimal solutions there are for the Hammersley TSP in Example 8.11.

8.14 Consider a complete graph with n nodes. With each edge from node i to j there is an associated cost c_{ij}. In the *longest path problem* the objective is to find the longest self-avoiding path from a certain *source* node to a *sink* node.

a) Assuming the source node is 1 and the sink node is n, formulate the longest path problem similar to the TSP. (The main difference is that the paths in the longest path problem can have different lengths.)

b) Specify a path generation mechanism and the corresponding CE updating rules.

c) Implement a CE algorithm for the longest path problem and apply it to a test problem.

8.15 Write a CE program that solves the eight-queens problem using the same configuration representation $\mathbf{X} = (X_1, \ldots, X_8)$ as in Example 6.13. A straightforward way to generate the configurations is to draw each X_i independently from a probability vector $(p_{i1}, \ldots, p_{i8})$, $i = 1, \ldots, 8$. Take $N = 500$, $\alpha = 0.7$, and $\varrho = 0.1$.

8.16 In the *permutation flow shop problem* (PFSP) n jobs have to be processed (in the same order) on m machines. The objective is to find the permutation of jobs that will minimize the *makespan*, that is, the time at which the last job is completed on machine m. Let $t(i, j)$ be the processing time for job i on machine j and let $\mathbf{x} = (x_1, x_2, ..., x_n)$ be a job permutation. Then the completion time $C(x_i, j)$ for job i on machine j can be calculated as follows:

$$\begin{aligned} C(x_1, 1) &= t(x_1, 1) \\ C(x_i, 1) &= C(x_{i-1}, 1) + t(x_i, 1), \forall i = 2, ..., n \\ C(x_1, j) &= C(x_1, j-1) + t(x_1, j), \forall j = 2, ..., m \\ C(x_i, j) &= \max\{C(x_{i-1}, j), C(x_i, j-1)\} + t(x_i, j)\ , \\ &\quad \text{for all } i = 2, ..., n; \quad j = 2, ..., m\ . \end{aligned}$$

The objective is to minimize $S(\mathbf{x}) = C(x_n, m)$. The trajectory generation for the PFSP is similar to that of the TSP.

a) Implement a CE algorithm to solve this problem.

b) Run the algorithm for a benchmark problem from the Internet, for example `http://ina2.eivd.ch/Collaborateurs/etd/problemes.dir/ordonnancement.dir/ordonnancement.html`.

8.17 Verify the updating formulas (8.46) and (8.47).

8.18 Plot Matlab's `peaks` function and verify that it has three local maxima.

8.19 Use the CE program in Section A.5 of the Appendix to maximize the function $S(x) = \mathrm{e}^{-(x-2)^2} + 0.8\,\mathrm{e}^{-(x+2)^2}$. Examine the convergence of the algorithm by plotting in the same figure the sequence of normal sampling densities.

8.20 Use the CE method to minimize the *trigonometric function*

$$S(\mathbf{x}) = 1 + \sum_{i=1}^{n} 8\sin^2(\eta(x_i - x_i^*)^2) + 6\sin^2(2\eta(x_i - x_i^*)^2) + \mu(x_i - x_i^*)^2\ , \quad (8.50)$$

with $\eta = 7$, $\mu = 1$, and $x_i^* = 0.9, i = 1, \ldots, n$. The global minimum $\gamma^* = 1$ is attained at $\mathbf{x}^* = (0.9, \ldots, 0.9)$. Display the graph and density plot of this function and give a table for the evolution of the algorithm.

8.21 A well-known test case in continuous optimization is the *Rosenbrock* function (in n dimensions):

$$S(\mathbf{x}) = \sum_{i=1}^{n-1} 100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\ . \quad (8.51)$$

The function has a global minimum $\gamma^* = 0$, attained at $\mathbf{x}^* = (1, 1, \ldots, 1)$. Implement a CE algorithm to minimize this function for dimensions $n = 2, 5, 10$, and 20. Observe how injection (Remark 8.7.1) affects the accuracy and speed of the algorithm.

8.22 Suppose that $\mathscr{X}$ in (8.15) is a (possibly nonlinear) region defined by the following system of inequalities:

$$G_i(\mathbf{x}) \leqslant 0, \quad i = 1, \ldots, L \,. \tag{8.52}$$

The *proportional penalty* approach to constrained optimization is to modify the objective function as follows:

$$\widetilde{S}(\mathbf{x}) = S(\mathbf{x}) + \sum_{i=1}^{L} P_i(\mathbf{x}) \,, \tag{8.53}$$

where $P_i(\mathbf{x}) = C_i \max(G_i(\mathbf{x}), 0)$ and $C_i > 0$ measures the importance (cost) of the i-th penalty. It is clear that as soon as the constrained problem (8.15), (8.52) is reduced to the unconstrained one (8.15) — using (8.53) instead of S — we can again apply Algorithm 8.3.1.

Apply the proportional penalty approach to the constrained minimization of the Rosenbrock function of dimension 10 for the constraints below. List for each case the minimal value obtained by the CE algorithm (with injection, if necessary) and the CPU time. In all experiments, use $\varepsilon = 10^{-3}$ for the stopping criterion (stop if all standard deviations are less than ε) and $C = 1000$. Repeat the experiments 10 times to check if indeed a global minimum is found.

a) $\sum_{j=1}^{10} x_j \leqslant -8$

b) $\sum_{j=1}^{10} x_j \geqslant 15$

c) $\sum_{j=1}^{10} x_j \leqslant -8, \ \sum_{j=1}^{10} x_j^2 \geqslant 15$

d) $\sum_{j=1}^{10} x_j \geqslant 15, \ \sum_{j=1}^{10} x_j^2 \leqslant 22.5$

8.23 Use the CE method to minimize the function

$$S(\mathbf{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1 x_2 - x_1 x_3 \,,$$

subject to the constraints $x_j \geqslant 0, j = 1, 2, 3$, and

$$\begin{aligned} 8\,x_1 + 14\,x_2 + 7\,x_3 - 56 &= 0 \,, \\ x_1^2 + x_2^2 + x_3^2 - 25 &= 0 \,. \end{aligned}$$

First, eliminate two of the variables by expressing x_2 and x_3 in terms of x_1. Note that this gives *two* different expressions for the pair (x_2, x_3). In the CE algorithm, generate the samples $\mathbf{X}$ by first drawing X_1 according to a truncated normal distribution on [0,5]. Then choose either the first or the second expression for (X_2, X_3) with equal probability. Verify that the optimal solution is approximately $\mathbf{x}^* = (3.51, 0.217, 3.55)$, with $S(\mathbf{x}^*) = 961.7$. Give the solution and the optimal value in seven significant digits.

8.24 Add $\mathsf{U}(-0.1, 0.1)$, $\mathsf{N}(0, 0.01)$, and $\mathsf{N}(0, 1)$ noise to the objective function in Problem 8.19. Formulate an appropriate stopping criterion, for example based on $\widehat{\sigma}_t$. For each case, observe how $\widehat{\gamma}_t$, $\widehat{\mu}_t$, and $\widehat{\sigma}_t$ behave.

8.25 Add $\mathsf{N}(0, 1)$ noise to the Matlab `peaks` function and apply the CE algorithm to find the global maximum. Display the contour plot and the path followed by the mean vectors $\{\widehat{\boldsymbol{\mu}}_t\}$, starting with $\widehat{\boldsymbol{\mu}}_0 = (1.3, -2.7)$ and using $N = 200$ and $\varrho = 0.1$. Stop when all standard deviations are less than $\varepsilon = 10^{-3}$. In a separate plot, display the evolution of the worst and best of the elite samples ($\widehat{\gamma}_t$ and S_t^*) at each iteration of the CE algorithm. In addition, evaluate and plot the noisy objective function in $\widehat{\boldsymbol{\mu}}_t$ for each iteration. Observe that in contrast to the deterministic case, the $\{\widehat{\gamma}_t\}$ and $\{S_t^*\}$ do not converge to γ^* because

of the noise, but eventually $S(\widehat{\boldsymbol{\mu}}_t)$ fluctuates around the optimum γ^*. More importantly, observe that the means $\{\widehat{\boldsymbol{\mu}}_t\}$ do converge to the optimal $\mathbf{x}^*$.

8.26 Select a particular instance (cost matrix) of the synthetic TSP in Problem 8.10. Make this TSP *noisy* by defining the random cost Y_{ij} from i to j in (8.48) to be $\mathsf{Exp}(c_{ij}^{-1})$ distributed. Apply the CE Algorithm 8.3.1 to the noisy problem and compare the results with those in the deterministic case. Display the evolution of the algorithm in a graph, plotting the maximum distance, $\max_{i,j} |\widehat{p}_{t,ij} - p_{ij}^*|$, as a function of t.

Further Reading

The CE method was pioneered in [26] as an adaptive algorithm for estimating probabilities of rare events in complex stochastic networks. Originally it was based on variance minimization. It was soon realized [27, 28] that the same technique (using CE rather than VM) could be used not only for estimation but also for optimization purposes.

A gentle tutorial on the CE method is given in [8] and a more comprehensive treatment can be found in [31]. In 2005 a whole volume (134) of the *Annals of Operations Research* was devoted to the CE method. The CE home page, featuring many links, articles, references, tutorials, and computer programs on CE, can be found at

`http://www.cemethod.org`

The CE method has applications in many areas, including buffer allocation [1], queueing models of telecommunication systems [7, 9], control and navigation [10], signal detection [18], DNA sequence alignment [12], scheduling and vehicle routing [4], reinforcement learning [19, 22], project management [5] and heavy-tail distributions [2], [16]. Applications to more classical combinatorial optimization problems are given in [28], [29], and [30]. The continuous counterpart is discussed in [15], and applications to clustering analysis are given in [3] and [17]. Various CE estimation and noisy optimization problems for reliability systems and network design can be found in [11], [13], [14], [23], [24], and [25]. Convergence issues are discussed in [6], [21], and Section 3.5 of [31].

An approach closely related to CE is the *probability collectives* work of Dr. David Wolpert and his collaborators. This approach uses information theory as a bridge to relate game theory, statistical physics, and distributed optimization; see, for example, [33, 34].

REFERENCES

1. G. Alon, D. P. Kroese, T. Raviv, and R. Y. Rubinstein. Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Annals of Operations Research*, 134:137–151, 2005.
2. S. Asmussen, D. P. Kroese, and R. Y. Rubinstein. Heavy tails, importance sampling and cross-entropy. *Stochastic Models*, 21(1):57–76, 2005.
3. Z. Botev and D. P. Kroese. Global likelihood optimization via the cross-entropy method, with an application to mixture models. In R. G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, editors, *Proceedings of the 2004 Winter Simulation Conference*, pages 529–535, Washington, DC, December 2004.
4. K. Chepuri and T. Homem de Mello. Solving the vehicle routing problem with stochastic demands using the cross entropy method. *Annals of Operations Research*, 134(1):153–181, 2005.

5. I. Cohen, B. Golany, and A. Shtub. Managing stochastic finite capacity multi-project systems through the cross-entropy method. *Annals of Operations Research*, 134(1):183–199, 2005.

6. A. Costa, J. Owen, and D. P. Kroese. Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters*, 35(5):573–580, 2007.

7. P. T. de Boer. *Analysis and Effi cient Simulation of Queueing Models of Telecommunication Systems*. PhD thesis, University of Twente, 2000.

8. P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

9. P. T. de Boer, D. P. Kroese, and R. Y. Rubinstein. A fast cross-entropy method for estimating buffer overflows in queueing networks. *Management Science*, 50(7):883–895, 2004.

10. B. E. Helvik and O. Wittner. Using the cross-entropy method to guide/govern mobile agent's path finding in networks. In S. Pierre and R. Glitho, editors, *Mobile Agents for Telecommunication Applications: Third International Workshop, MATA 2001, Montreal*, pages 255–268, New York, 2001. Springer-Verlag.

11. K-P. Hui, N. Bean, M. Kraetzl, and D.P. Kroese. The cross-entropy method for network reliability estimation. *Annals of Operations Research*, 134:101–118, 2005.

12. J. Keith and D. P. Kroese. Sequence alignment by rare event simulation. In *Proceedings of the 2002 Winter Simulation Conference*, pages 320–327, San Diego, 2002.

13. D. P. Kroese and K. P. Hui. In: *Computational Intelligence in Reliability Engineering*, chapter 3: Applications of the Cross-Entropy Method in Reliability. Springer-Verlag, New York, 2006.

14. D. P. Kroese, S. Nariai, and K. P. Hui. Network reliability optimization via the cross-entropy method. *IEEE Transactions on Reliability*, 56(2):275–287, 2007.

15. D. P. Kroese, S. Porotsky, and R. Y. Rubinstein. The cross-entropy method for continuous multi-extremal optimization. *Methodology and Computing in Applied Probability*, 2006.

16. D. P. Kroese and R. Y. Rubinstein. The transform likelihood ratio method for rare event simulation with heavy tails. *Queueing Systems*, 46:317–351, 2004.

17. D. P. Kroese, R. Y. Rubinstein, and T. Taimre. Application of the cross-entropy method to clustering and vector quantization. *Journal of Global Optimization*, 37:137–157, 2007.

18. Z. Liu, A. Doucet, and S. S. Singh. The cross-entropy method for blind multiuser detection. In *IEEE International Symposium on Information Theory*, Chicago, 2004. Piscataway.

19. S. Mannor, R. Y. Rubinstein, and Y. Gat. The cross-entropy method for fast policy search. In *The 20th International Conference on Machine Learning (ICML-2003)*, Washington, DC, 2003.

20. L. Margolin. *Cross-Entropy Method for Combinatorial Optimization*. Master's thesis, The Technion, Israel Institute of Technology, Haifa, July 2002.

21. L. Margolin. On the convergence of the cross-entropy method. *Annals of Operations Research*, 134(1):201–214, 2005.

22. I. Menache, S. Mannor, and N. Shimkin. Basis function adaption in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238, 2005.

23. S. Nariai and D. P. Kroese. On the design of multi-type networks via the cross-entropy method. In *Proceedings of the Fifth International Workshop on the Design of Reliable Communication Networks (DRCN)*, pages 109–114, 2005.

24. S. Nariai, D. P. Kroese, and K. P. Hui. Designing an optimal network using the cross-entropy method. In *Intelligent Data Engineering and Automated Learning*, Lecture Notes in Computer Science, pages 228–233, New York, 2005. Springer-Verlag.

25. A. Ridder. Importance sampling simulations of Markovian reliability systems using cross-entropy. *Annals of Operations Research*, 134(1):119–136, 2005.

26. R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99:89–112, 1997.

27. R. Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 2:127–190, 1999.

28. R. Y. Rubinstein. Combinatorial optimization, cross-entropy, ants and rare events. In S. Uryasev and P. M. Pardalos, editors, *Stochastic Optimization: Algorithms and Applications*, pages 304–358, Dordrecht, 2001. Kluwer.

29. R. Y. Rubinstein. Combinatorial optimization via cross-entropy. In S. Gass and C. Harris, editors, *Encyclopedia of Operations Research and Management Sciences*, pages 102–106. Kluwer, 2001.

30. R. Y. Rubinstein. The cross-entropy method and rare-events for maximal cut and bipartition problems. *ACM Transactions on Modelling and Computer Simulation*, 12(1):27–53, 2002.

31. R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unifi ed Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*. Springer-Verlag, New York, 2004.

32. R. Y. Rubinstein and B. Melamed. *Modern Simulation and Modeling*. John Wiley & Sons, New York, 1998.

33. D. H. Wolpert. Information theory: the bridge connecting bounded rational game theory and statistical physics. In D. Braha and Y. Bar-Yam, editors, *Complex Engineering Systems*. Perseus Books, 2004.

34. D. H. Wolpert and S. R. Bieniawski. Distributed control by Lagrangian steepest descent. In *IEEE Conference on Decision and Control*, volume 2, pages 1562–1567, 2004.

This Page Intentionally Left Blank

CHAPTER 9

COUNTING VIA MONTE CARLO

9.1 COUNTING PROBLEMS

Many important problems in science, engineering, and mathematics deal with counting problems that are *#P-complete* [22, 23, 30]. This is a concept related to the familiar class of NP-hard problems. Here are some examples of #P-complete counting problems:

- The *Hamiltonian cycle* counting problem. How many Hamiltonian cycles exist in a given graph? That is, how many different tours (cycles) does the graph contain that visit each node (vertex) exactly once, apart for the beginning/end node? Note that the problem of finding a particular Hamiltonian cycle is a special case of the TSP in which the distances between adjacent nodes are 1, and other distances are 0, and the objective is to find the *longest* tour.

- The *self-avoiding walk* problem. How many walks are there of a certain length n that, starting from the origin, move between the nearest grid points and do not visit the same grid point more than once?

- The *satisfiability* (SAT) counting problem. Let $\{x_1, \ldots, x_n\}$ be a collection of n Boolean variables. Let C be a set of Boolean clauses. Examples of such clauses are $x_1 \vee \bar{x}_2$ and $(x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2)$. How many (if any) satisfying truth assignments for C exist? That is, how many ways are there to set the variables $x_1, \ldots, x_n$ either true or false so that each clause in C is true?

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

Some other #P-complete problems include counting the number of perfect matches in a bipartite graph, determining the permanent of a matrix, counting the number of fixed-size cliques in a graph, and counting the number of forests in a graph.

It is interesting to note [23, 30] that in many cases the counting problem is hard to solve, while the associated decision or optimization problem is easy; in other words, *decision is easy, counting is hard.* For example, finding the shortest path between two fixed vertices in a graph is easy, while finding the total number of paths between the two vertices is difficult.

In this chapter we show how #P-complete counting problems can be viewed as particular instances of *estimation* problems, and as such can be solved efficiently using Monte Carlo techniques, such as importance sampling and MCMC. We also show that when using the standard CE method to estimate adaptively the optimal importance sampling density, one can encounter degeneracy in the likelihood ratio, which leads to highly variable estimates for large-dimensional problems. We solve this problem by introducing a particular modification of the classic *MinxEnt* method [17], called the *parametric MinxEnt* (PME) method. We show that PME is able to overcome the *curse of the dimensionality* (degeneracy) of the likelihood ratio by decomposing it into low-dimensional parts. Much of the theory is illustrated via the satisfiability counting problem in the conjunctive normal form (CNF), which plays a central role in NP completeness. We also present here a different approach, which is based on sequential sampling. The idea is to break a difficult counting problem into a combination of easier ones. In particular, for the SAT problem in the disjunctive normal form (DNF), we design an importance sampling algorithm and show that it possesses nice complexity properties.

Although #P-complete problems, and in particular SAT, are of both theoretical and practical importance and have been well studied for at least a quarter of a century, we are not aware of any generic deterministic or randomized method for *fast* counting for such problems. We are not even aware of any benchmark problems to which our method can be compared. One goal of this chapter is therefore to motivate more research and applications on #P-complete problems, as the original CE method did in the fields of Monte Carlo simulation and simulation-based optimization in recent years.

The rest of this chapter is organized as follows. Section 9.2 introduces the SAT counting problem. In Section 9.3 we show how a counting problem can be reduced to a rare-event estimation one. In Section 9.4 we consider a sequential sampling plan, where a difficult counting problem $|\mathscr{X}^*|$ can be presented as a combination of associated easy ones. Based on the above sequential sampling we design an efficient importance sampling algorithm. We show that for the SAT problem in the DNF form the proposed algorithm possesses nice complexity properties. Section 9.5 deals with SAT counting in the CNF form, using the rare-event approach developed in Section 9.3. In particular, we design an algorithm, called the PME algorithm, which is based on a combination of importance sampling and the classic MinxEnt method. In Section 9.6 we show that the PME method can be applied to combinatorial optimization problems as well and can be viewed as an alternative to the standard CE method. The efficiency of the PME method is demonstrated numerically in Section 9.7. In particular, we show that PME works at least as well as the standard CE for combinatorial optimization problems and substantially outperforms the latter for SAT counting problems.

9.2 SATISFIABILITY PROBLEM

The Boolean satisfiability (SAT) problem plays a central role in combinatorial optimization and, in particular, in NP completeness. Any NP-complete problem, such as the max-cut

problem, the graph-coloring problem, and the TSP, can be translated *in polynomial time* into a SAT problem. The SAT problem plays a central role in solving large-scale computational problems, such as planning and scheduling, integrated circuit design, computer architecture design, computer graphics, image processing, and finding the folding state of a protein.

There are different formulations for the SAT problem, but the most common one, which we discuss next, consists of two components [12]:

- A set of n Boolean variables $\{x_1, \ldots, x_n\}$, representing statements that can either be TRUE (=1) or FALSE (=0). The negation (the logical NOT) of a variable x is denoted by $\overline{x}$. For example, $\overline{\texttt{TRUE}} = \texttt{FALSE}$. A variable or its negation is called a *literal.*
- A set of m distinct *clauses* $\{C_1, C_2, \ldots, C_m\}$ of the form $C_i = z_{i_1} \vee z_{i_2} \vee \cdots \vee z_{i_k}$, where the z's are literals and the $\vee$ denotes the logical OR operator. For example, $0 \vee 1 = 1$.

The binary vector $\mathbf{x} = (x_1, \ldots, x_n)$ is called a *truth assignment*, or simply an *assignment.* Thus, $x_i = 1$ assigns truth to x_i and $x_i = 0$ assigns truth to $\overline{x}_i$ for each $i = 1, \ldots, n$. The simplest SAT problem can now be formulated as: *find a truth assignment* $\mathbf{x}$ *such that* all *clauses are true.*

Denoting the logical AND operator by $\wedge$, we can represent the above SAT problem via a single formula as

$$F_1 = C_1 \wedge C_2 \wedge \cdots \wedge C_m \ ,$$

where the $\{C_k\}$ consist of literals connected with only $\vee$ operators. The SAT formula is said to be in *conjunctive normal form* (CNF). An alternative SAT formulation concerns formulas of the type

$$F_2 = C_1 \vee C_2 \vee \cdots \vee C_m \ ,$$

where the clauses are of the form $C_i = z_{i_1} \wedge z_{i_2} \wedge \cdots \wedge z_{i_k}$. Such a SAT problem is then said to be in *disjunctive normal form* (DNF). In this case, a truth assignment $\mathbf{x}$ is sought that satisfies *at least one* of the clauses, which is usually a much simpler problem.

■ **EXAMPLE 9.1**

As an illustration of the SAT problem and the corresponding SAT counting problem, consider the following toy example of coloring the nodes of the graph in Figure 9.1. Is it possible to color the nodes either black or white in such a way that no two adjacent nodes have the same color? If so, how many such colorings are there?

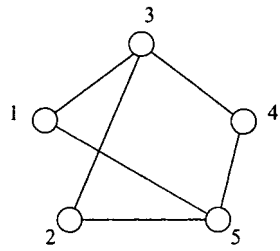


Figure 9.1 Can the graph be colored with two colors so that no two adjacent nodes have the same color?

We can translate this graph coloring problem into a SAT problem in the following way: Let x_j be the Boolean variable representing the statement "the j-th node is colored black". Obviously, x_j is either TRUE or FALSE, and we wish to assign truth to either x_j or $\overline{x}_j$, for each $j = 1, \ldots, 5$. The restriction that adjacent nodes cannot have the same color can be translated into a number of clauses that must all hold. For example, "node 1 and node 3 cannot both be black" can be translated as clause $C_1 = \overline{x}_1 \vee \overline{x}_3$. Similarly, the statement "at least one of node 1 and node 2 must be black" is translated as $C_2 = x_1 \vee x_3$. The same holds for all other pairs of adjacent nodes. The clauses can now be conveniently summarized as in Table 9.1. Here, in the left-hand table, for each clause C_i a 1 in column j means that the clause contains x_j, a -1 means that the clause contains the negation $\overline{x}_j$; and a 0 means that the clause does not contain either of them. Let us call the corresponding matrix $A = (a_{ij})$ the *clause matrix*. For example, $a_{75} = -1$ and $a_{42} = 0$. An alternative representation of the clause matrix is to list for each clause only the indices of all Boolean variables present in that clause. In addition, each index that corresponds to a negation of a variable is preceded by a minus sign; see Table 9.1.

Table 9.1 A SAT table and an alternative representation of the clause matrix.

	1	2	3	4	5	
x	0	1	0	1	0	
C_1	-1	0	-1	0	0	1
C_2	1	0	1	0	0	0
C_3	-1	0	0	0	-1	1
C_4	1	0	0	0	1	0
C_5	0	-1	-1	0	0	1
C_6	0	1	1	0	0	1
C_7	0	-1	0	0	-1	1
C_8	0	1	0	0	1	1
C_9	0	0	-1	-1	0	1
C_{10}	0	0	1	1	0	1
C_{11}	0	0	0	-1	-1	1
C_{12}	0	0	0	1	1	1

C_1	-1	-3
C_2	1	3
C_3	-1	-5
C_4	1	5
C_5	-2	-3
C_6	2	3
C_7	-2	-5
C_8	2	5
C_9	-3	-4
C_{10}	3	4
C_{11}	-4	-5
C_{12}	4	5

Now let $\mathbf{x} = (x_1, \ldots, x_5)$ be a truth assignment. The question is whether there exists an $\mathbf{x}$ such that all clauses $\{C_k\}$ are satisfied. To see if a single clause C_k is satisfied, one must compare the truth assignment for each variable in that clause with the values $1, -1$, and 0 in the clause matrix A, which indicates if the literal corresponds to the variable, its negation, or that neither appears in the clause. If, for example, $x_j = 0$ and $a_{ij} = -1$, then the literal $\overline{x}_j$ is TRUE. The entire clause is TRUE if it contains at least one true literal. Define the clause value $C_i(\mathbf{x}) = 1$ if clause C_i is TRUE with truth assignment $\mathbf{x}$ and $C_i(\mathbf{x}) = 0$ if it is FALSE. Then it is easy to see that

$$C_i(\mathbf{x}) = \max_j \{0,\ (2\,x_j - 1)\,a_{ij}\}\ , \tag{9.1}$$

assuming that at least one a_{ij} is nonzero for clause C_i (otherwise, the clause can be deleted). For example, for truth assignment $(0, 1, 0, 1, 0)$ the corresponding clause values are given in the rightmost column of the lefthand table in Table 9.1. We see that

the second and fourth clauses are violated. However, the assignment $(1, 1, 0, 1, 0)$ does indeed yield all clauses true, and this therefore gives a way in which the nodes can be colored: 1 = black, 2 = black, 3 = white, 4 = black, 5 = white. It is easy to see that $(0, 0, 1, 0, 1)$ is the only other assignment that renders all the clauses true.

The problem of deciding whether there *exists* a valid assignment, and indeed providing such a vector, is called the *SAT-assignment* problem [21]. Finding a coloring in Example 9.1 is a particular instance of the SAT assignment problem. A SAT-assignment problem in which each clause contains exactly K literals is called a *K-SAT* problem. It is well known that 2-SAT problems are easy (can be solved in polynomial time), while K-SAT problems for $K \geq 3$ are NP-hard. A more difficult problem is to find the *maximum* number of clauses that can be satisfied by one truth assignment. This is called the *MAX-SAT* problem. Recall that our ultimate goal is counting rather than decision making, that is, to find *how many* truth assignments exist that satisfy a given set of clauses.

9.2.1 Random K-SAT (K-RSAT)

Although K-SAT counting problems for $K \geqslant 2$ are NP-hard, numerical studies nevertheless indicate that most K-SAT problems are easy to solve for certain values of n and m. To study this phenomena, Mézard and Montanari [21] define a family of *random* K-SAT problems, which we denote by K-RSAT(m, n). Each instance of a K-RSAT(m, n) contains m clauses of length K corresponding to n variables. Each clause is drawn uniformly from the set of $\binom{n}{K} 2^K$ clauses, independently of the other clauses. It has been observed empirically that a crucial parameter characterizing this problem is

$$\beta = \frac{m}{n}, \tag{9.2}$$

which is called the *clause density*.

Denote by $P(n, K, \beta)$ the probability that a randomly generated SAT instance is satisfiable. Figure 9.2, adapted from [11], shows $P(n, 3, \beta)$ as a function of β for $n = 50, 100$, and 200 (the larger the n, the steeper the curve).

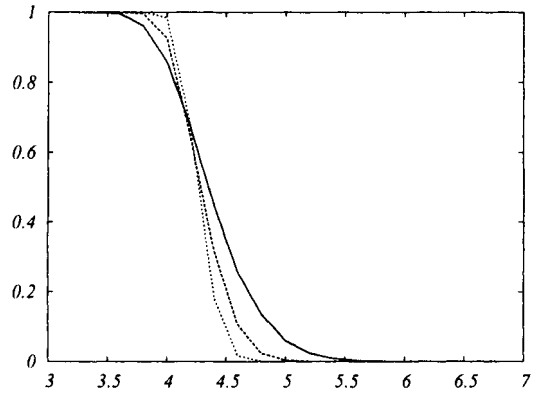


Figure 9.2 The probability that a K-RSAT(m, n) problem has a solution as a function of the clause density β for $n = 50, 100$, and 200.

One can see that for fixed n this is a decreasing function of β. It starts from 1 for $\beta = 0$ and goes to 0 as β goes to infinity. An interesting observation from these simulation studies is the existence of a *phase transition* at some finite value β^*, in the sense that for $\beta < \beta^*$ K-RSAT(m, n) is satisfied with probability $P(n, K, \beta) \to 1$ as $n \to \infty$, while for $\beta > \beta^*$ the same probability goes to 0 as $n \to \infty$. For example, it has been found empirically that $\beta^* \approx 4.26$ for $K = 3$. Similar behavior of $P(n, K, \beta)$ has been observed for other values of K. In particular, it has been found empirically that for fixed n, β^* increases in K and the crossover from high to low probabilities becomes sharper and sharper as n increases. Moreover, it is proved rigorously in [21] that

1. For 2-RSAT$(n\beta, n)$: $\lim_{n\to\infty} P(n, 2, \beta) = \begin{cases} 1 & \text{if } \beta < 1, \\ 0 & \text{if } \beta > 1. \end{cases}$

2. For K-RSAT$(n\beta, n)$, $K \geq 3$, there exist a $\beta^* = \beta^*(K)$, such that

$$\lim_{n\to\infty} P(n, K, \beta) = \begin{cases} 1 & \text{if } \beta < \beta^*, \\ 0 & \text{if } \beta > \beta^*. \end{cases}$$

Finally, It has been shown empirically in [21] that for fixed n and K the computational effort needed to solve the random K-SAT problem has a peak at the vicinity of the point β^* and the value of the peak increases rapidly in n.

One thus distinguishes the following three regions for K-RSAT$(n\beta, n)$:

1. For small β, the problem of finding a solution is easy and the CPU time grows polynomial in n.

2. At the phase transition region (near β^*), the problem (finding a solution or showing that a solution does not exist) becomes hard and the CPU time typically grows exponential in n.

3. For $\beta > \beta^*$ the problem becomes easier but still requires exponential time. In this region there is likely to be no solution; the objective is therefore to show efficiently that the problem is UNSAT.

It follows that hardest instances of the random SAT are located around the phase transition region (the vicinity of β^*). In our numerical studies below, we shall present the performance of the PME algorithm for such hard instances while treating the SAT counting problem.

9.3 THE RARE-EVENT FRAMEWORK FOR COUNTING

We start with the fundamentals of the Monte Carlo method for estimation and counting by considering the following basic example.

EXAMPLE 9.2

Suppose we want to calculate an area of some irregular region $\mathscr{X}^*$. The Monte Carlo method suggests inserting the irregular region $\mathscr{X}^*$ into a nice regular one $\mathscr{X}$, say a rectangle (see Figure 9.3), and then applying the following estimation procedure:

1. Generate a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *uniformly* distributed over the regular region $\mathscr{X}$.

2. Estimate the desired area $|\mathscr{X}^*|$ as

$$\widehat{|\mathscr{X}^*|} = |\mathscr{X}| \frac{1}{N} \sum_{k=1}^{N} I_{\{\mathbf{X}_k \in \mathscr{X}^*\}} , \tag{9.3}$$

where $I_{\{\mathbf{X}_k \in \mathscr{X}^*\}}$ denotes the indicator of the event $\{\mathbf{X}_k \in \mathscr{X}^*\}$. Note that according to (9.3) we accept the generated point $\mathbf{X}_k$ if $\mathbf{X}_k \in \mathscr{X}^*$ and reject it otherwise.

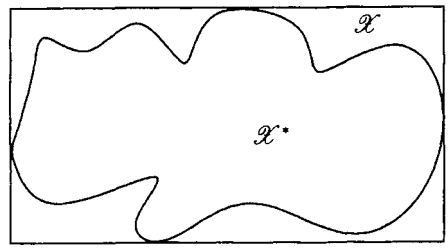


Figure 9.3 Illustration of the acceptance-rejection method.

Formula (9.3) is also valid for *counting problems*, that is, when $\mathscr{X}^*$ is a discrete rather than a continuous set of points. In this case, one generates a uniform sample over the grid points of some larger nice region $\mathscr{X}^*$ and then, as before, uses the acceptance-rejection method to estimate $|\mathscr{X}^*|$.

Since in most interesting counting problems $\{\mathbf{X}_k \in \mathscr{X}^*\}$ is a rare event we shall use importance sampling, because the acceptance–rejection method is meaningless in this case. Let g be an importance sampling pdf defined on some set $\mathscr{X}$ and let $\mathscr{X}^* \subset \mathscr{X}$; then $|\mathscr{X}^*|$ can be written as

$$|\mathscr{X}^*| = \sum_{\mathbf{x} \in \mathscr{X}} I_{\{\mathbf{x} \in \mathscr{X}^*\}} \frac{g(\mathbf{x})}{g(\mathbf{x})} = \mathbb{E}_g \left[\frac{I_{\{\mathbf{x} \in \mathscr{X}^*\}}}{g(\mathbf{X})} \right] . \tag{9.4}$$

To estimate $|\mathscr{X}^*|$ via Monte Carlo, we draw a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from g and take the estimator

$$\widehat{|\mathscr{X}^*|} = \frac{1}{N} \sum_{k=1}^{N} I_{\{\mathbf{X}_k \in \mathscr{X}^*\}} \frac{1}{g(\mathbf{X}_k)} . \tag{9.5}$$

The best choice for g is $g^*(\mathbf{x}) = 1/|\mathscr{X}^*|$, $\mathbf{x} \in \mathscr{X}^*$; in other words, $g^*(\mathbf{x})$ is the uniform pdf over the discrete set $\mathscr{X}^*$. Under g^* the estimator has zero variance, so that only *one sample is required.* Clearly, such g^* is infeasible. However, for various counting problems a natural choice for g presents itself, as illustrated in the following example.

■ EXAMPLE 9.3 Self-Avoiding Walk

The self-avoiding random walk, or simply *self-avoiding walk*, is a basic mathematical model for polymer chains. For simplicity we shall deal only with the two-dimensional

case. Each self-avoiding walk is represented by a path $\mathbf{x} = (x_1, x_2, \ldots, x_{n-1}, x_n)$, where x_i represents the two-dimensional position of the $i-$th molecule of the polymer chain. The distance between adjacent molecules is fixed at 1, and the main requirement is that the chain does not self-intersect. We assume that the walk starts at the origin. An example of a self-avoiding walk walk of length 130 is given in Figure 9.4.

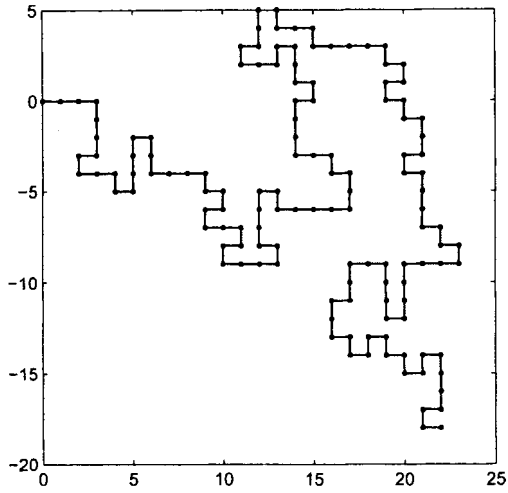


Figure 9.4 A self-avoiding random walk of length $n = 130$.

One of the main questions regarding the self-avoiding walk model is: how many self-avoiding walks are there of length n? Let $\mathscr{X}^*$ be the set of self-avoiding walks of length n. We wish to estimate $|\mathscr{X}^*|$ via (9.5) by employing a convenient pdf $g(\mathbf{x})$. This pdf is defined by the following *one-step-look-ahead* procedure.

Procedure (One-Step-Look-Ahead)

1. Let $X_0 = (0,0)$. Set $t = 1$.
2. Let d_t be the number of neighbors of X_{t-1} that have not yet been visited. If $d_t > 0$, choose X_t with probability $1/d_t$ from its neighbors. If $d_t = 0$, stop generating the path.
3. Stop if $t = n$. Otherwise, increase t by 1 and go to Step 2.

Note that the procedure generates either a self-avoiding walk $\mathbf{x}$ of length n or a part thereof. Let $g(\mathbf{x})$ be the corresponding discrete pdf. Then, for any self-avoiding walk $\mathbf{x}$ of length n, we have by the product rule (1.4)

$$g(\mathbf{x}) = \frac{1}{d_1}\frac{1}{d_2}\cdots\frac{1}{d_n} = \frac{1}{w(\mathbf{x})},$$

where

$$w(\mathbf{x}) = d_1 \cdots d_n . \tag{9.6}$$

The self-avoiding walk counting algorithm now follows directly from (9.5).

Algorithm 9.3.1 (Counting Self-Avoiding Walks)

1. *Generate independently N paths $\mathbf{X}_1, \ldots, \mathbf{X}_N$ via the one-step-look-ahead procedure.*

2. *For each self-avoiding walk $\mathbf{X}_k$, compute the corresponding $w(\mathbf{X}_k)$ as in* (9.6). *For the other paths, set $w(\mathbf{X}_k) = 0$.*

3. *Return*

$$\widehat{|\mathscr{X}^*|} = \frac{1}{N} \sum_{i=k}^{N} w(\mathbf{X}_k) \,.$$

The efficiency of the simple one-step-look-ahead method deteriorates rapidly as n becomes large. It becomes impractical to simulate walks of length more than 200. This is due to the fact that if at any one step t the point x_{t-1} does not have unoccupied neighbors ($d_t = 0$), then the "weight" $w(\mathbf{x})$ is zero and contributes nothing to the final estimate of $|\mathscr{X}^*|$. This problem can occur early in the simulation, rendering any subsequent sequential build-up useless. Better-performing algorithms do not restart from scratch but reuse successful partial walks to build new walks. These methods usually split the self avoiding partial walks into a number of copies and continue them as if they were independently built up from scratch. We refer to [20] for a discussion of these more advanced algorithms.

In general, choosing the importance sampling pdf g close to g^* to yield a good (low-variance) estimator for $|\mathscr{X}^*|$ may not be straightforward. However, there are several different approaches for constructing such low-variance pdfs. Among them are the standard CE, *exponential change of measure* (ECM), and the celebrated *MinxEnt* method [17]. Here we shall us a particular modification of the MinxEnt method called the *PME method* and show numerically that for the SAT problem it outperforms substantially the standard CE approach.

9.3.1 Rare Events for the Satisfiability Problem

Next, we demonstrate how to reduce the calculation of the number of SAT assignments to the estimation of rare-event probabilities. Let $A = (a_{ij})$ be a general $m \times n$ clause matrix representing the variables or negations thereof that occur in the clauses. Consider, for example, the 3×5 clause matrix in Table 9.2.

Table 9.2 A clause matrix with five clauses for three variables.

−1	1	0
−1	1	1
−1	1	−1
−1	−1	1
1	1	−1

Thus, $a_{ik} = 1$ and $a_{ik} = -1$ correspond to literals and negations, respectively; the 0 in cell (1,3) means that neither the third variable nor its negation occur at clause C_1. For any truth assignment $\mathbf{x} = (x_1, \ldots, n)$, let $C_i(\mathbf{x})$ be 1 if the i-th clause is TRUE for assignment $\mathbf{x}$ and 0 otherwise, $i = 1, \ldots, m$. Thus, the $C_i(\mathbf{x})$ can be computed via (9.1). Next, define

$$S(\mathbf{x}) = \sum_{i=1}^{m} C_i(\mathbf{x}) \; . \tag{9.7}$$

Table 9.3 presents the eight possible assignment vectors and the corresponding values of $S(\mathbf{x})$ for the clause matrix in Table 9.2.

Table 9.3 The eight assignment vectors and the corresponding values of $S(\mathbf{x})$.

x_1	x_2	x_3	$S(\mathbf{x})$
0	0	0	5
0	0	1	4
0	1	0	5
0	1	1	5
1	0	0	3
1	0	1	3
1	1	0	4
1	1	1	5

Recall that our goal is to find, for a given set of n Boolean variables and a set of m clauses, *how many* truth assignments exist that satisfy all the clauses. If we call the set of all 2^n truth assignments $\mathscr{X}$ and denote the subset of those assignments that satisfy all clauses by $\mathscr{X}^*$, then our objective is to count $|\mathscr{X}^*|$. It is readily seen from Table 9.3 that the clauses are simultaneously satisfied for four assignments, each corresponding to $S(\mathbf{x}) = 5$. Thus, in this case $|\mathscr{X}^*| = 4$.

The connection with rare-event simulation is the following. Let

$$\ell = \frac{|\mathscr{X}^*|}{|\mathscr{X}|} = \mathbb{P}_{\mathbf{p}_U}(\mathbf{X} \in \mathscr{X}^*) = \mathbb{P}_{\mathbf{p}_U}(S(\mathbf{X}) = m) \; , \tag{9.8}$$

where $\mathbf{p}_U$ denotes the "uniform" probability vector $(1/2, \ldots, 1/2)$. In other words, ℓ in (9.8) is the probability that a uniformly generated SAT assignment (trajectory) $\mathbf{X}$ is valid, that is, all clauses are satisfied, which is typically very small. We have thus reduced the SAT counting problem to a problem involving the estimation of a rare-event probability, and we can proceed directly with updating the probability vector $\mathbf{p}$ in order to estimate efficiently the probability ℓ, and thus also the number of valid trajectories $|\mathscr{X}^*|$.

9.4 OTHER RANDOMIZED ALGORITHMS FOR COUNTING

In the previous section we explained how Monte Carlo algorithms can be used for counting using the importance sampling estimator (9.5). In this section we look at some alternatives. In particular, we consider a sequential sampling plan, where the difficult problem of counting $|\mathscr{X}^*|$ is decomposed into "easy" problems of counting the number of elements in a sequence

of related sets $\mathscr{X}_1, \ldots, \mathscr{X}_m$. A typical procedure for such a decomposition can be written as follows:

1. Formulate the counting problem as the problem of estimating the cardinality of some set $\mathscr{X}^*$.

2. Find sets $\mathscr{X}_0, \mathscr{X}_1, \ldots, \mathscr{X}_m$ such that $|\mathscr{X}_m| = |\mathscr{X}^*|$ and $|\mathscr{X}_0|$ is known.

3. Write $|\mathscr{X}^*| = |\mathscr{X}_m|$ as

$$|\mathscr{X}^*| = |\mathscr{X}_0| \prod_{j=1}^{m} \frac{|\mathscr{X}_j|}{|\mathscr{X}_{j-1}|} . \tag{9.9}$$

4. Develop an efficient estimator $\widehat{\eta}_j$ for each $\eta_j = |\mathscr{X}_j|/|\mathscr{X}_{j-1}|$, resulting in an efficient estimator,

$$\widehat{|\mathscr{X}^*|} = |\mathscr{X}_0| \prod_{j=1}^{m} \widehat{\eta}_j , \tag{9.10}$$

for $|\mathscr{X}^*|$.

Algorithms such as based on the sequential sampling estimator (9.10) are sometimes called *randomized algorithms* in the computer literature [22]. We will refer to the notion of a randomized algorithm as an algorithm that introduces randomness during its execution. In particular, the standard CE and the PME algorithm below can be viewed as examples of randomized algorithms.

Remark 9.4.1 (Uniform Sampling) Finding an efficient estimator for each $\eta_j = |\mathscr{X}_j|/|\mathscr{X}_{j-1}|$ is the crux of the counting problem. A very simple and powerful idea is to obtain such an estimator by sampling *uniformly* from the set $\mathscr{Y}_j = \mathscr{X}_{j-1} \cup \mathscr{X}_j$. By doing so, one can simply take the proportion of samples from $\mathscr{Y}_j$ that fall in $\mathscr{X}_j$ as the estimator for η_j. For such an estimator to be efficient (have low variance), the subset $\mathscr{X}_j$ must be relatively "dense" in $\mathscr{Y}_j$. In other words η_j should not be too small.

If exact sampling from the uniform distribution on some set $\mathscr{Y}$ is difficult or impossible, one can resort to *approximate* sampling, for example via the Metropolis–Hastings Algorithm 6.2.1; see in particular Example 6.2.

It is shown in [22] and [23] that many interesting counting problems can be put into the setting (9.9). In fact, the CNF SAT counting problem in Section 9.3.1 can be formulated in this way. Here the objective is to estimate $|\mathscr{X}^*| = |\mathscr{X}|\,|\mathscr{X}^*|/|\mathscr{X}| = |\mathscr{X}|\,\ell$, where $|\mathscr{X}|$ is known and ℓ can be estimated via importance sampling. Below we give some more examples.

■ **EXAMPLE 9.4 Independent Sets**

Consider a graph $G = (V, E)$ with m edges and n vertices. Our goal is to count the number of independent node (vertex) sets of the graph. A node set is called *independent* if no two nodes are connected by an edge, that is, if no two nodes are adjacent; see Figure 9.5 for an illustration of this concept.

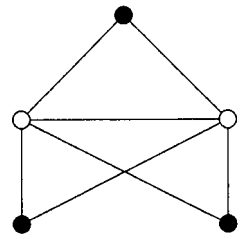


Figure 9.5 The black nodes form an independent set, since they are not adjacent to each other.

Consider an arbitrary ordering of the edges. Let E_j be the set of the first j edges and let $G_j = (V, E_j)$ be the associated subgraph. Note that $G_m = G$ and that G_j is obtained from G_{j+1} by removing an edge. Denoting $\mathscr{X}_j$ the set of independent sets of G_j, we can write $|\mathscr{X}^*| = |\mathscr{X}_m|$ in the form (9.9). Here $|\mathscr{X}_0| = 2^n$, since G_0 has no edges and thus every subset of V is an independent set, including the empty set. Note that here $\mathscr{X}_0 \supset \mathscr{X}_1 \supset \cdots \supset \mathscr{X}_m = \mathscr{X}^*$.

■ EXAMPLE 9.5 Knapsack Problem

Given items of sizes $a_1, \ldots, a_m > 0$ and a positive integer $b \geqslant \min_i a_i$, find the number of vectors $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ such that

$$\sum_{i=1}^{n} a_i x_i \leq b \,.$$

The integer b represents the size of the knapsack, and x_i indicates whether or not item i is put in the knapsack. Let $\mathscr{X}^*$ denote the set of all feasible solutions, that is, all different combinations of items that can be placed into the knapsack without exceeding its size. The goal is to determine $|\mathscr{X}^*|$.

To put the knapsack problem into the framework (9.9), assume without loss of generality that $a_1 \leqslant a_2 \leqslant \cdots \leqslant a_n$ and define $b_j = \sum_{i=1}^{j} a_i$, with $b_0 = 0$. Denote $\mathscr{X}_j$ the set of vectors $\mathbf{x}$ that satisfy $\sum_{i=1}^{n} a_i x_i \leqslant b_j$, and let m be the largest integer such that $b_m \leqslant b$. Clearly, $\mathscr{X}_m = \mathscr{X}^*$. Thus, (9.9) is established again.

■ EXAMPLE 9.6 Counting the Permanent

The permanent of a general $n \times n$ matrix $A = (a_{ij})$ is defined as

$$\operatorname{per}(A) = |\mathscr{X}^*| = \sum_{\mathbf{x} \in \mathscr{X}} \prod_{i=1}^{n} a_{ix_i} \,, \tag{9.11}$$

where $\mathscr{X}$ is the set of all permutations $\mathbf{x} = (x_1, \ldots, x_n)$ of $(1, \ldots, n)$. It is well known that the calculation of the permanent of a *binary* matrix is equivalent to the calculation of the number of perfect matchings in a certain bipartite graph. A *bipartite graph* $G = (V, E)$ is a graph in which the node set V is the union of two disjoint sets V_1 and V_2, and in which each edge joins a node in V_1 to a node in V_2. A *matching* of size m is a collection of m edges in which each node occurs at most once. A *perfect matching* is a matching of size n.

To see the relation between the permanent of a binary matrix $A = (a_{ij})$ and the number of perfect matchings in a graph, consider the bipartite graph G, where V_1 and V_2 are disjoint copies of $\{1, \ldots, n\}$ and $(i, j) \in E$ if and only if $a_{ij} = 1$ for all i and j. As an example, let A be the 3×3 matrix

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}. \tag{9.12}$$

The corresponding bipartite graph is given in Figure 9.6. The graph has three perfect matchings, one of which is displayed in the figure. These correspond to all permutations $\mathbf{x}$ for which the product $\prod_{i=1}^{n} a_{ix_i}$ is equal to 1.

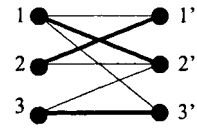


Figure 9.6 A bipartite graph. The bold edges form a perfect matching.

For a general binary $(n \times n)$ matrix A, let $\mathscr{X}_j$ denote the set of matchings of size j in the corresponding bipartite graph G. Assume that $\mathscr{X}_n$ is nonempty, so that G has a perfect matching of nodes V_1 and V_2. We are interested in calculating $|\mathscr{X}_n| = \text{per}(A)$. Taking into account that $|\mathscr{X}_1| = |E|$, we obtain the product form (9.9).

As a final application of (9.9), we consider the general problem of counting the number of elements in the *union* of some sets $\mathscr{X}_1, \ldots, \mathscr{X}_m$.

9.4.1 $\mathscr{X}^*$ is a Union of Some Sets

Let, as usual, $\mathscr{X}$ be a finite set of objects, and let $\mathscr{X}^*$ denote a subset of special objects that we wish to count. In specific applications $\mathscr{X}^*$ frequently can be written as the *union* of some sets $\mathscr{X}_1, \ldots, \mathscr{X}_m$, as illustrated in Figure 9.7.

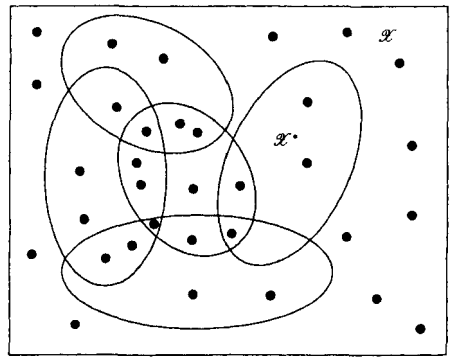


Figure 9.7 Count the number of points in the gray set $\mathscr{X}^*$.

As a special case we shall consider the counting problem for a SAT in DNF. Recall that a DNF formula is a disjunction (logical OR) of clauses $C_1 \vee C_2 \vee \cdots \vee C_m$, where each clause is a conjunction (logical AND) of literals. Let $\mathscr{X}$ be the set of all assignments, and let $\mathscr{X}_j$ be the subset of all assignments satisfying clause C_j, $j = 1, \ldots, m$. Denote by $\mathscr{X}^*$ the set of assignments satisfying *at least one* of the clauses $C_1, \ldots, C_m$, that is, $\mathscr{X}^* = \cup_{j=1}^m \mathscr{X}_j$. The DNF counting problem is to compute $|\mathscr{X}^*|$. It is readily seen that if a clause C_j has n_j literals, then the number of true assignments is $|\mathscr{X}_j| = 2^{n-n_j}$. Clearly, $0 \leq |\mathscr{X}^*| \leq |\mathscr{X}| = 2^n$ and, because an assignment can satisfy more than one clause, also $|\mathscr{X}^*| \leqslant \sum_{j=1}^m |\mathscr{X}_j|$.

Next, we shall show how to construct a randomized algorithm for this #P-complete problem. The first step is to *augment* the state space $\mathscr{X}$ with an index set $\{1, \ldots, m\}$. Specifically, define

$$\mathscr{A} = \{(j, \mathbf{x}) : \mathbf{x} \in \mathscr{X}_j,\ j = 1, \ldots, m\} . \tag{9.13}$$

This set is illustrated in Figure 9.8. In this case we have $m = 7$, $|\mathscr{X}_1| = 3$, $|\mathscr{X}_2| = 2$, and so on.

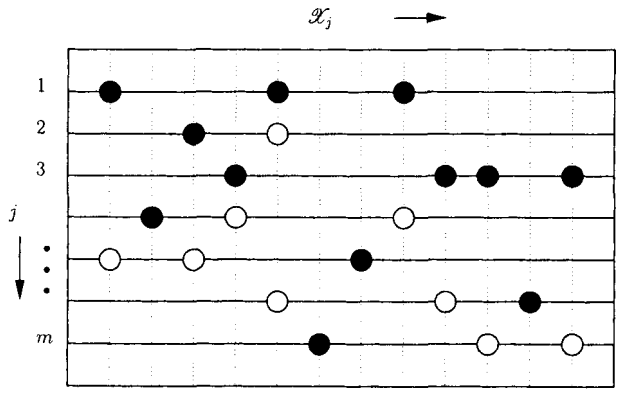


Figure 9.8 The sets $\mathscr{A}$ (formed by all points) and $\mathscr{A}^*$ (formed by the black points).

For a fixed j we can identify the subset $\mathscr{A}_j = \{(j, \mathbf{x}) : \mathbf{x} \in \mathscr{X}_j\}$ of $\mathscr{A}$ with the set $\mathscr{X}_j$. In particular, the two sets have the same number of elements. Next, we construct a subset $\mathscr{A}^*$ of $\mathscr{A}$ with size exactly equal to $|\mathscr{X}^*|$. This is done by associating with each assignment in $\mathscr{X}^*$ exactly one pair $(j, \mathbf{x})$ in $\mathscr{A}$. In particular, we can use the pair with the smallest clause index number, that is, we can define $\mathscr{A}^*$ as

$$\mathscr{A}^* = \{(j, \mathbf{x}) : \mathbf{x} \in \mathscr{X}_j,\ \mathbf{x} \notin \mathscr{X}_k \text{ for } k < j,\ j = 1, \ldots, m\} .$$

In Figure 9.8 $\mathscr{A}^*$ is represented by the black points. Note that each element of $\mathscr{X}^*$ is represented once in $\mathscr{A}$, that is, each "column" has exactly one black point.

Since $|\mathscr{A}| = \sum_{j=1}^m |\mathscr{X}_j| = \sum_{j=1}^m 2^{n-n_j}$ is available, we can estimate $|\mathscr{X}^*| = |\mathscr{A}^*| = |\mathscr{A}|\ \ell$ by estimating $\ell = |\mathscr{A}^*|/|\mathscr{A}|$. Note that this is a simple application of (9.9). The ratio ℓ can be estimated by generating pairs *uniformly* in $\mathscr{A}$ and counting how often they occur in $\mathscr{A}^*$. It turns out that for the union of sets, and in particular for the DNF problem, generating pairs uniformly in $\mathscr{A}$ is quite straightforward and can be done in two stages using

the composition method. Namely, first choose an index j, $j = 1, \ldots, m$ with probability

$$p_j = \frac{|\mathscr{A}_j|}{|\mathscr{A}|} = \frac{|\mathscr{X}_j|}{\sum_{j=1}^m |\mathscr{X}_j|} = \frac{2^{n-n_j}}{\sum_{j=1}^m 2^{n-n_j}} ;$$

next, choose an assignment $\mathbf{x}$ uniformly from $\mathscr{X}_j$. This can be done by choosing a value 1 or 0 with equal probability and independently for each literal that is *not in clause* j. The resulting probability of choosing the pair $(j, \mathbf{x})$ can be found via conditioning as

$$\mathbb{P}(J = j, \mathbf{X} = \mathbf{x}) = \mathbb{P}(J = j)\,\mathbb{P}(\mathbf{X} = \mathbf{x} \mid J = j) = \frac{|\mathscr{A}_j|}{|\mathscr{A}|} \frac{1}{|\mathscr{A}_j|} = \frac{1}{|\mathscr{A}|} ,$$

which corresponds to the uniform distribution on $\mathscr{A}$. The DNF counting algorithm can be written as follows [22]

Algorithm 9.4.1 (DNF Counting Algorithm)

Given is a DNF formula with m clauses and n literals.

1. *Let $Z = 0$.*
2. *For $k = 1$ to N:*
   i. *With probability $p_j \propto |\mathscr{X}_j|$, choose uniformly and randomly an assignment $\mathbf{X} \in \mathscr{X}_j$.*
   ii. *If $\mathbf{X}$ is not in any $\mathscr{X}_i$ for $i < j$, increase Z by 1.*
3. *Return*

$$\widehat{|\mathscr{X}^*|} = \frac{Z}{N} \sum_{j=1}^m |\mathscr{X}_j| \tag{9.14}$$

as the estimate of the number $|\mathscr{X}^*|$ of satisfying assignments.

Note that the ratio $\ell = |\mathscr{A}^*|/|\mathscr{A}|$ can be written as

$$\ell = \mathbb{E}_{\mathsf{U}} \left[I_{\{\mathbf{A} \in \mathscr{A}^*\}} \right] , \tag{9.15}$$

where the subscript U indicates that $\mathbf{A}$ is drawn uniformly over $\mathscr{A}$. Algorithm 9.4.1 counts the quantity $\frac{Z}{N}$ (an estimator of ℓ), representing the ratio of the number of accepted samples Z to the total generated N, and then it multiplies $\frac{Z}{N}$ by the constant $\sum_{j=1}^m |\mathscr{X}_j| = |\mathscr{A}|$. Note also that Algorithm 9.4.1 can be applied to some other problems involving the union of quite arbitrary sets $\mathscr{X}_j, j = 1, \ldots, m$.

Next, we present an alternative estimation procedure for ℓ in (9.15). Conditioning on $\mathbf{X}$, we obtain by the conditioning property (1.11):

$$\ell = \mathbb{E}_{\mathsf{U}}[p(\mathbf{X})] ,$$

where $p(\mathbf{X}) = \mathbb{P}_{\mathsf{U}}(I_{\{\mathbf{A} \in \mathscr{A}^*\}} \mid \mathbf{X})$ is the conditional probability that a uniformly chosen $\mathbf{A} = (J, \mathbf{X})$ falls in set $\mathscr{A}^*$, given $\mathbf{X}$. For a given element $\mathbf{x} \in \mathscr{X}^*$, let $r(\mathbf{x})$ denote the number of sets $\mathscr{X}_j$ to which $\mathbf{x}$ belongs. For example, in Figure 9.8 the values for $r(\mathbf{x})$ from left to right are 2, 1, 2, 2, 3, 1,.... Given a particular $\mathbf{x}$, the probability that the corresponding $(J, \mathbf{x})$ lies in $\mathscr{A}^*$ — in the figure, this means that the corresponding point in the column corresponding to $\mathbf{x}$ is black — is simply $1/r(\mathbf{x})$, because each of the $r(\mathbf{x})$ points

is chosen uniformly and there is only one representative of $\mathscr{A}^*$ in each column. In other words, $p(\mathbf{X}) = 1/r(\mathbf{X})$. Hence, if $r(\mathbf{x})$ can be calculated for each $\mathbf{x}$, one can estimate $\ell = \mathbb{E}_U[1/r(\mathbf{X})]$ as Y/N, with $Y = \sum_{k=1}^{N} \frac{1}{r(\mathbf{X}_k)}$. By doing so, we obtain the estimator

$$\widetilde{|\mathscr{X}^*|} = \frac{Y}{N} \sum_{j=1}^{m} |\mathscr{X}_j| \,. \tag{9.16}$$

Note that in contrast to (9.14) the estimator in (9.16) avoids the acceptance-rejection step. Both $\widehat{|\mathscr{X}^*|}$ and $\widetilde{|\mathscr{X}^*|}$ are unbiased estimators of $|\mathscr{X}^*|$, but the latter has the smaller variance of the two, because it is obtained by conditioning; see the conditional Monte Carlo Algorithm 5.4.1.

Both $\widehat{|\mathscr{X}^*|}$ and $\widetilde{|\mathscr{X}^*|}$ can be viewed as importance sampling estimators of the form (9.4). We shall show it for the latter. Namely, let $g(\mathbf{x}) = r(\mathbf{x})/c$, $\mathbf{x} \in \mathscr{X}^*$, where c is a normalization constant, that is, $c = \sum_{\mathbf{x}\in\mathscr{X}^*} r(\mathbf{x}) = \sum_{i=1}^{m} |\mathscr{X}_i|$. Then, applying (9.5), with $\mathscr{A}^*$ and $\mathscr{A}$ instead of $\mathscr{X}^*$ and $\mathscr{X}$, gives the estimator

$$\sum_{k=1}^{N} \frac{1}{g(\mathbf{X}_k)} = \sum_{i=1}^{m} |\mathscr{X}_i| \sum_{k=1}^{N} \frac{1}{r(\mathbf{X}_k)} \,,$$

which is exactly $\widetilde{|\mathscr{X}^*|}$. As mentioned, sampling from the importance sampling pdf $g(\mathbf{x})$ is done via the composition method without explicitly resorting to $r(\mathbf{x})$. Namely, by selecting $(J, \mathbf{X})$ uniformly over $\mathscr{A}$, we have

$$\mathbb{P}(\mathbf{X} = \mathbf{x}) = \frac{r(\mathbf{x})}{|\mathscr{A}|} = g(\mathbf{x}), \; \mathbf{x} \in \mathscr{X}^* \,.$$

We shall show below that the DNF Counting Algorithm 9.4.1 possesses some nice complexity properties.

9.4.2 Complexity of Randomized Algorithms: FPRAS and FPAUS

A randomized algorithm is said to give an (ε, δ)-*approximation* of a parameter z if its output Z satisfies

$$\mathbb{P}(|Z - z| \leqslant \varepsilon z) \geq 1 - \delta \,, \tag{9.17}$$

that is, the "relative error" $|Z - z|/z$ of the approximation Z lies with high probability $(> 1 - \delta)$ below some small number ε.

One of the main tools in proving (9.17) for various randomized algorithms is the so-called *Chernoff bound*, which states that for any random variable Y and any number a

$$\mathbb{P}(Y \leqslant a) \leqslant \min_{\theta>0} \mathrm{e}^{\theta a} \, \mathbb{E}[\mathrm{e}^{-\theta Y}] \,. \tag{9.18}$$

Namely, for any fixed a and $\theta > 0$, define the functions $H_1(z) = I_{\{z \leqslant a\}}$ and $H_2(z) = \mathrm{e}^{\theta(a-z)}$. Then, clearly, $H_1(z) \leqslant H_2(z)$ for all z. As a consequence, for any θ,

$$\mathbb{P}(Y \leqslant a) = \mathbb{E}[H_1(Y)] \leqslant \mathbb{E}[H_2(Y)] = \mathrm{e}^{\theta a} \, \mathbb{E}[\mathrm{e}^{-\theta Y}] \,.$$

The bound (9.18) now follows by taking the smallest such θ. An important application is the following.

Theorem 9.4.1 *Let $X_1, \ldots, X_n$ be iid* Ber(p) *random variables. Then their sample mean provides an (ε, δ)-approximation for p, that is,*

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - p\right| \leqslant \varepsilon p\right) \geq 1 - \delta\,, \tag{9.19}$$

provided that $n \geq 3\ln(2/\delta)/(p\varepsilon^2)$.

Proof: Let $Y = X_1 + \cdots + X_n$, and $\ell_L = \mathbb{P}(Y \leqslant (1-\varepsilon)np)$. Because $\mathbb{E}[\mathrm{e}^{-\theta Y}] = \mathbb{E}[\mathrm{e}^{-\theta X_1}]^n = (1 - p + p\,\mathrm{e}^{\theta})^n$, the Chernoff bound gives

$$\ell_L \leqslant \mathrm{e}^{\theta np(1-\varepsilon)}\,(1 - p + p\,\mathrm{e}^{\theta})^n\,, \tag{9.20}$$

for any $\theta > 0$. By direct differentiation we find that the optimal θ^* (giving the smallest upper bound) is

$$\theta^* = \ln\left(\frac{\varepsilon p - p + 1}{(\varepsilon - 1)(p-1)}\right).$$

It is not difficult to verify (see Problem 9.1) that by substituting $\theta = \theta^*$ in the right-hand side of (9.20) and taking the logarithms on both sides, $\ln(\ell_L)$ can be upper-bounded by $np\,h(p,\varepsilon)$, where $h(p,\varepsilon)$ is given by

$$h(\varepsilon, p) = -\frac{1}{p}\ln\left(1 + \frac{\varepsilon p}{1-p}\right) + (1-\varepsilon)\theta^*\,. \tag{9.21}$$

For fixed $0 < \varepsilon < 1$, the function $h(p,\varepsilon)$ is monotonically decreasing in p, $0 < p < 1$. Namely,

$$\frac{\partial h(\varepsilon,p)}{\partial p} = \frac{1}{p^2}\left[-\frac{\varepsilon p}{1-p} + \ln\left(1 + \frac{\varepsilon p}{1-p}\right)\right] < 0\,,$$

since $-y + \ln(1+y) < 0$ for any $y > 0$. It follows that

$$h(\varepsilon, p) \leqslant h(\varepsilon, 0) = -\varepsilon - (1-\varepsilon)\ln(1-\varepsilon) = -\sum_{k=2}^{\infty}\frac{\varepsilon^k}{(k-1)k} \leqslant \frac{-\varepsilon^2}{2}\,.$$

And therefore,

$$\ell_L \leqslant \exp\left(-\frac{np\varepsilon^2}{2}\right).$$

Similarly, Chernoff's bound provides the following upper bound for $\ell_U = \mathbb{P}(Y \geqslant (1+\varepsilon)np) = \mathbb{P}(-Y \leqslant -(1+\varepsilon)np)$:

$$\ell_U \leqslant \exp\left(-\frac{np\varepsilon^2}{2 + \frac{2}{3}\varepsilon}\right) \tag{9.22}$$

for all $0 < \varepsilon < 1$; see Problem 9.2. In particular, $\ell_L + \ell_U \leqslant 2\exp(-np\varepsilon^2/3)$. Combining these results gives

$$\mathbb{P}(|Y - np| \leqslant np\varepsilon) = 1 - \ell_L - \ell_U \geqslant 1 - 2\mathrm{e}^{-np\,\varepsilon^2/3},$$

so that by choosing $n \geqslant 3\ln(2/\delta)/(p\varepsilon^2)$, the above probability is guaranteed to be greater than or equal to $1 - \delta$. □

Definition 9.4.1 (FPRAS) A randomized algorithm is said to provide a *fully polynomial randomized approximation scheme (FPRAS)* if, for any input vector $\mathbf{x}$ and any parameters $\varepsilon > 0$ and $0 < \delta < 1$, the algorithm outputs an (ε, δ)-approximation to the desired quantity $z(\mathbf{x})$ in time that is polynomial in $\varepsilon^{-1}, \ln \delta^{-1}$ and the size n of the input vector $\mathbf{x}$.

Thus, the sample mean in Theorem 9.4.1 provides an FPRAS for estimating p. Note that the input vector $\mathbf{x}$ consists of the Bernoulli variables $X_1, \ldots, X_n$.

Below we present a theorem [22] stating that Algorithm 9.4.1 provides an FPRAS for counting the number of satisfying assignments in a DNF formula. Its proof is based on the fact that $\mathscr{A}^*$ is relatively *dense* in $\mathscr{A}$. Specifically, it uses the fact that for the union of sets $\ell = |\mathscr{A}^*|/|\mathscr{A}| \geq 1/m$, which follows directly from the fact that each assignment can satisfy at most m clauses.

Theorem 9.4.2 (DNF Counting Theorem) *The DNF counting Algorithm 9.4.1 is an FPRAS, provided that* $N \geq 3m \ln(2/\delta)/\varepsilon^2$.

Proof: Step 2 of Algorithm 9.4.1 chooses an element uniformly from $\mathscr{A}$. The probability that this element belongs to $\mathscr{A}^*$ is at least $1/m$. Choose

$$N = \frac{3m}{\varepsilon^2} \ln \frac{2}{\delta}\,, \tag{9.23}$$

where $\varepsilon > 0$ and $\delta > 0$. Then N is polynomial in m, ε^{-1} and $\ln \frac{1}{\delta}$, and the processing time of each sample is polynomial in m. By Theorem 9.4.1 we find that with the number of samples N as in (9.23), the quantity Z/N (see Algorithm 9.4.1) provides an (ε, δ)-approximation to ℓ and thus $\widehat{|\mathscr{X}^*|}$ provides an (ε, δ)-approximation to $|\mathscr{X}^*|$.

□

As observed at the beginning of this section, there exists a fundamental connection between *uniform sampling* from some set $\mathscr{X}$ (such as the set $\mathscr{A}$ for the DNF counting problem) and *counting* the number of elements of interest in this set [1, 22]. Since, as we mentioned, exact uniform sampling is not always feasible, MCMC techniques are often used to sample *approximately* from a uniform distribution.

Let Z be the random output of a sampling algorithm on a finite sample space $\mathscr{X}$. We say that the sampling algorithm generates an *ε-uniform sample* from $\mathscr{X}$ if, for any $\mathscr{Y} \subset \mathscr{X}$,

$$\left| \mathbb{P}(Z \in \mathscr{Y}) - \frac{|\mathscr{Y}|}{|\mathscr{X}|} \right| \leqslant \varepsilon\,. \tag{9.24}$$

Definition 9.4.2 (FPAUS) A sampling algorithm is called a *fully polynomial almost uniform sampler (FPAUS)* if, given an input vector $\mathbf{x}$ and a parameter $\varepsilon > 0$, the algorithm generates an ε-uniform sample from $\mathscr{X}(\mathbf{x})$ and runs in time that is polynomial in $\ln \varepsilon^{-1}$ and the size of the input vector $\mathbf{x}$.

■ EXAMPLE 9.7 FPAUS for Independent Sets: Example 9.4 Continued

An FPAUS for independent sets takes as input a graph $G = (V, E)$ and a parameter $\varepsilon > 0$. The sample space $\mathscr{X}$ consists of all independent sets in G, with the output being an ε-uniform sample from $\mathscr{X}$. The time required to produce such an ε-uniform sample should be polynomial in the size of the graph and $\ln \varepsilon^{-1}$. The final goal is to prove that given an FPAUS, one can construct a corresponding FPRAS. Such a proof is based on the product formula (9.9) and is given in Theorem 10.5 of [22].

For the knapsack problem, it can be shown that there is an FPRAS *provided* that there exists an FPAUS; see also Exercise 10.6 of [22]. However, the existence of such a method is still an open problem [15].

9.4.3 FPRAS for SATs in CNF

Next, we shall show that all the above results obtained so far for SATs in the DNF also apply to SATs in the CNF. In particular, the proof of FPRAS and FPAUS is simple and therefore quite surprising. It is based on De Morgan's law,

$$\left(\bigcap \mathscr{X}_i\right)^c = \bigcup \mathscr{X}_i^c \quad \text{and} \quad \left(\bigcup \mathscr{X}_i\right)^c = \bigcap \mathscr{X}_i^c \,. \tag{9.25}$$

Thus, if the $\{\mathscr{X}_i\}$ are subsets of some set $\mathscr{X}$, then

$$\left|\bigcap \mathscr{X}_i\right| = |\mathscr{X}| - \left|\bigcup \mathscr{X}_i^c\right| \,. \tag{9.26}$$

In particular, consider a CNF SAT counting problem and let $\mathscr{X}_i$ be the set of all assignments that satisfy the i-th clause, C_i, $i = 1, \ldots, m$. Recall that C_i is of the form $z_{i1} \vee z_{i2} \vee \cdots \vee z_{ik}$. The set of assignments satisfying *all* clauses is $\mathscr{X}^* = \cap \mathscr{X}_i$. In view of (9.26), to count $\mathscr{X}^*$ one could instead count the number of elements in $\cup \mathscr{X}_i^c$. Now $\mathscr{X}_i^c$ is the set of all assignments that satisfy the clause $\overline{z}_{i1} \wedge \overline{z}_{i2} \wedge \cdots \wedge \overline{z}_{ik}$. Thus, the problem is translated into a DNF SAT counting problem.

As an example, consider the CNF SAT counting problem with clause matrix

$$A = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & -1 \end{pmatrix} \,.$$

In this case $\mathscr{X}^*$ comprises three assignments, namely, $(1,0,0)$, $(1,1,0)$, and $(1,1,1)$. Consider next the DNF SAT counting problem with clause matrix $-A$. Then the set of assignments that satisfy at least one clause for this problem is $\{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,1)\}$, which is exactly the complement of $\mathscr{X}^*$.

Since the DNF SAT counting problem can be solved via an FPRAS, and any CNF SAT counting problem can be straightforwardly translated into the former problem, an FPRAS can be derived for the CNF SAT counting problem.

9.5 MINXENT AND PARAMETRIC MINXENT

This section deals with the *parametric MinxEnt* (PME) method for estimating rare-event probabilities and counting, which is based on the MinxEnt method. Below we present some background on the MinxEnt method.

9.5.1 The MinxEnt Method

In the standard CE method for rare-event simulation, the importance sampling density for estimating $\ell = \mathbb{P}_f(S(\mathbf{X}) \geqslant \gamma)$ is restricted to some parametric family, say $\{f(\cdot;\mathbf{v}),\ \mathbf{v} \in \mathscr{V}\}$, and the optimal density $f(\cdot;\mathbf{v}^*)$ is found as the solution to the *parametric* CE minimization program (8.3). In contrast to CE, we present below a *nonparametric* method called the *MinxEnt* method. The idea is to minimize the CE distance to g^* over *all* pdfs rather than over $\{f(\cdot;\mathbf{v}),\ \mathbf{v} \in \mathscr{V}\}$. However, the program $\min_g \mathscr{D}(g, g^*)$ is void of meaning, since the minimum (zero) is attained at the unknown $g = g^*$. A more useful approach is to first specify a *prior* density h, which conveys the available information on the "target" g^*, and then choose the "instrumental" pdf g as close as possible to h, subject to certain *constraints*

on g. If no prior information on the target g^* is known, the prior h is simply taken to be a constant, corresponding to the uniform distribution (continuous or discrete). This leads to the following minimization framework [2], [3], and [17]:

$$(\mathrm{P}_0)\quad \begin{cases} \min_g \mathcal{D}(g,h) = \min_g \int \ln \frac{g(\mathbf{x})}{h(\mathbf{x})}\, g(\mathbf{x})\, \mathrm{d}\mathbf{x} = \min_g \mathbb{E}_g \left[\ln \frac{g(\mathbf{X})}{h(\mathbf{X})}\right] \\ \text{s.t.} \quad \int S_i(\mathbf{x})\, g(\mathbf{x})\, \mathrm{d}\mathbf{x} = \mathbb{E}_g[S_i(\mathbf{X})] = \gamma_i, \quad i = 1, \ldots, m\,, \\ \int g(\mathbf{x})\, \mathrm{d}\mathbf{x} = 1\,. \end{cases} \tag{9.27}$$

Here g and h are n-dimensional pdfs, $S_i(\mathbf{x})$, $i = 1, \ldots, m$, are given functions, and $\mathbf{x}$ is an n-dimensional vector. The program (P_0) is Kullback's *minimum cross-entropy* (MinxEnt) program. Note that this is a *convex* functional optimization problem, because the objective function is a convex function of g, and the constraints are affine in g.

If the prior h is constant, then $\mathcal{D}(g,h) = \int g(\mathbf{x}) \ln g(\mathbf{x})\, \mathrm{d}\mathbf{x}$ + constant, so that the minimization of $\mathcal{D}(g,h)$ in (P_0) can be replaced with the *maximization* of

$$\mathcal{H}(g) = -\int g(\mathbf{x}) \ln g(\mathbf{x})\, \mathrm{d}\mathbf{x} = -\mathbb{E}_g[\ln g(\mathbf{X})]\,, \tag{9.28}$$

where $\mathcal{H}(g)$ is the *Shannon entropy*; see (1.52). (Here we use a different notation to emphasize the dependence on g.) The corresponding program is Jaynes' *MaxEnt* program [13]. Note that the former minimizes the Kullback-Leibler cross-entropy, while the latter maximizes the Shannon entropy [17].

In typical counting and combinatorial optimization problems h is chosen as an n-dimensional pdf with uniformly distributed marginals. For example, in the SAT counting problem, we assume that each component of the n-dimensional random vector $\mathbf{X}$ is $\mathsf{Ber}(1/2)$ distributed. While estimating rare events in stochastic models, like queueing models, h is the fixed underlying pdf. For example, in the $M/M/1$ queue h would be a two-dimensional pdf with independent marginals, where the first marginal is the interarrival $\mathsf{Exp}(\lambda)$ pdf and the second is the service $\mathsf{Exp}(\mu)$ pdf.

The MinxEnt program, which presents a constrained functional optimization problem, can be solved via Lagrange multipliers. The solution for the discrete case is derived in Example 1.20 on page 39. A similar solution can be derived, for example , via calculus of variations [2], for the general case. In particular, the solution of the MinxEnt problem is given [17] by

$$g(\mathbf{x}) = \frac{h(\mathbf{x}) \exp\left\{\sum_{i=1}^m \lambda_i\, S_i(\mathbf{x})\right\}}{\mathbb{E}_h\left[\exp\left\{\sum_{i=1}^m \lambda_i\, S_i(\mathbf{X})\right\}\right]}\,, \tag{9.29}$$

where λ_i, $i = 1, \ldots, m$ are obtained from the solution of the following system of equations:

$$\frac{\mathbb{E}_h\left[S_i(\mathbf{X}) \exp\left\{\sum_{j=1}^m \lambda_j\, S_j(\mathbf{X})\right\}\right]}{\mathbb{E}_h\left[\exp\left\{\sum_{j=1}^m \lambda_j\, S_j(\mathbf{X})\right\}\right]} = \gamma_i, \quad i = 1, \ldots, m\,, \tag{9.30}$$

where $\mathbf{X} \sim h(\mathbf{x})$.

Note that $g(\mathbf{x})$ can be written as

$$g(\mathbf{x}) = C(\boldsymbol{\lambda})\, h(\mathbf{x})\, \exp\left\{\sum_{i=1}^m \lambda_i\, S_i(\mathbf{x})\right\}, \tag{9.31}$$

where

$$C^{-1}(\boldsymbol{\lambda}) = \mathbb{E}_h \left[\exp \left\{ \sum_{i=1}^{m} \lambda_i S_i(\mathbf{X}) \right\} \right] \tag{9.32}$$

is the normalization constant. Note also that $g(\mathbf{x})$ is a density function; in particular, $g(\mathbf{x}) \geq 0$.

Consider the MinxEnt program (P_0) with a single constraint, that is,

$$\begin{cases} \min_g \mathcal{D}(g,h) = \min_g \mathbb{E}_g \left[\ln \frac{g(X)}{h(X)} \right] \\ \text{s.t.} \quad \mathbb{E}_g[S(X)] = \gamma \,, \\ \int g(x)\, \mathrm{d}x = 1 \end{cases} \tag{9.33}$$

In this case (9.29) and (9.30) reduce to

$$g(\mathbf{x}) = \frac{h(\mathbf{x})\, \exp\{\lambda\, S(\mathbf{x})\}}{\mathbb{E}_h \left[\exp\{\lambda\, S(\mathbf{X})\}\right]} \tag{9.34}$$

and

$$\frac{\mathbb{E}_h \left[S(\mathbf{X}) \exp\{\lambda\, S(\mathbf{X})\}\right]}{\mathbb{E}_h \left[\exp\{\lambda\, S(\mathbf{X})\}\right]} = \gamma \,, \tag{9.35}$$

respectively.

In the particular case where $S(\mathbf{x})$, $\mathbf{x} = (x_1, \ldots, x_n)$ is a coordinatewise separable function, that is,

$$S(\mathbf{x}) = \sum_{i=1}^{n} S_i(x_i) \tag{9.36}$$

and the components X_i, $i = 1, \ldots, n$ of the random vector $\mathbf{X}$ are independent under $h(\mathbf{x}) = h(x_1) \cdots h(x_n)$, the joint pdf $g(\mathbf{x})$ in (9.34) reduces to the *product of marginal pdfs*. In particular, the i-th component of $g(\mathbf{x})$ can be written as

$$g_i(x) = \frac{h_i(x) \exp\{\lambda S_i(x)\}}{\mathbb{E}_{h_i} \left[\exp\{\lambda S_i(x)\}\right]}, \quad i = 1, \ldots, n \,. \tag{9.37}$$

Remark 9.5.1 (The MinxEnt Program with Inequality Constraints) It is not difficult to extend the MinxEnt program to contain *inequality* constraints. Suppose that the following M inequality constraints are added to the MinxEnt program (9.27):

$$\mathbb{E}_g[S_i(\mathbf{X})] \geqslant \gamma_i, \qquad i = m+1, \ldots, m+M \,.$$

The solution of this MinxEnt program is given by

$$g(\mathbf{x}) = \frac{h(\mathbf{x}) \exp\left\{\sum_{i=1}^{m+M} \lambda_i\, S_i(\mathbf{x})\right\}}{\mathbb{E}_h \left[\exp\left\{\sum_{i=1}^{m+M} \lambda_i\, S_i(\mathbf{X})\right\}\right]} \,, \tag{9.38}$$

where the Lagrange multipliers $\lambda_1, \ldots, \lambda_{m+M}$ are the solutions to the dual convex optimization problem

$$\begin{aligned} \max_{\boldsymbol{\lambda}, \beta} \quad & \sum_{i=1}^{m+M} \lambda_i\, \gamma_i - \beta - \mathbb{E}_h \left[\exp\left(-1 - \beta + \sum_{i=1}^{m+M} \lambda_i S_i(\mathbf{x}) \right) \right] \\ \text{subject to:} \quad & \lambda_i \geqslant 0, \quad i = m+1, \ldots, m+M. \end{aligned}$$

Thus, only the Lagrange multipliers corresponding to an inequality must be constrained from below by zero. Similar to (1.87), this can be solved in two steps, where β can be determined explicitly as a normalization constant but the $\{\lambda_i\}$ have to be determined numerically.

In the special case of a single inequality constraint (that is, $m = 0$ and $M = 1$), the dual program can be solved directly (see also Problem 9.9), yielding the following solution:

$$\lambda = \begin{cases} 0 & \text{if } \mathbb{E}_h[S(\mathbf{X})] \geqslant \gamma \\ \lambda^* & \text{if } \mathbb{E}_h[S(\mathbf{X})] < \gamma \,, \end{cases}$$

where λ^* is obtained from (9.35). That is, if $\mathbb{E}_h[S(\mathbf{X})] < \gamma$, then the inequality MinxEnt solution agrees with the equality MinxEnt solution; otherwise, the optimal sampling pdf remains the prior h.

Remark 9.5.2 It is well known [9] that the optimal solution of the single-dimensional single-constrained MinxEnt program (9.33) coincides with the celebrated optimal *exponential change of measure* (ECM). Note that normally in a multidimensional ECM one twists each component separately, using possibly different twisting parameters. In contrast, the optimal solution to the MinxEnt program (see (9.37)) is parameterized by a *single-dimensional* parameter λ.

If not otherwise stated, we consider below only the single-constrained case (9.33). Like in the standard CE method one can also use a *multilevel* approach, where a sequence of instrumentals $\{g_t\}$ and levels $\{\gamma_t\}$ is used. Starting with $g_0 = f$ and always taking prior $h = f$, we determine γ_t and g_t as follows:

1. Update γ_t as
$$\gamma_t = \mathbb{E}_{g_t}[S(\mathbf{X}) \,|\, S(\mathbf{X}) \geqslant q_t] \,,$$
where q_t is the $(1-\varrho)$-quantile of $S(\mathbf{X})$ under g_{t-1}.
2. Update g_t as the solution to the above MinxEnt program for level γ_t rather than γ.

The updating formula for γ_t is based on the constraint $\mathbb{E}_g[S(\mathbf{X})] = \gamma$ in the MinxEnt program. However, instead of simply updating as $\gamma_t = \mathbb{E}_{g_{t-1}}[S(\mathbf{X})]$, we take the expectation of $S(\mathbf{X})$ with respect to g_{t-1} *conditional* upon $S(\mathbf{X})$ being greater than its $(1-\varrho)$ quantile, here denoted as q_t. In contrast, in the standard CE method the level γ_t is simply updated as q_t.

Note that each g_t is completely determined by its Lagrange multiplier, say λ_t, which is the solution to (9.35) with γ_t instead of γ. In practice, both γ_t and λ_t have to be replaced by their stochastic counterparts $\widehat{\gamma}_t$ and $\widehat{\lambda}_t$, respectively. Specifically, γ_t can be estimated from a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ of g_{t-1} as the average of the $N^{\mathrm{e}} = \lceil (1-\varrho)N \rceil$ elite sample performances:

$$\widehat{\gamma}_t = \frac{\sum_{i=N-N^{\mathrm{e}}+1}^{N} S_{(i)}}{N^{\mathrm{e}}} \,, \tag{9.39}$$

where $S_{(i)}$ denotes the i-th order-statistics of the sequence $S(\mathbf{X}_1), \ldots, S(\mathbf{X}_N)$. And λ_t can be estimated by solving, with respect to λ, the stochastic counterpart of (9.35), which is

$$\frac{\sum_{k=1}^{N} \mathrm{e}^{\lambda S(\mathbf{X}_k)} S(\mathbf{X}_k)}{\sum_{k=1}^{N} \mathrm{e}^{\lambda S(\mathbf{X}_k)}} = \widehat{\gamma}_t \,. \tag{9.40}$$

9.5.2 Rare-Event Probability Estimation Using PME

The above MinxEnt approach has limited applications [25], since it requires sampling from the complex multidimensional pdf $g(\mathbf{x})$ of $\mathbf{X}$. For this reason we shall consider in this section a modified version of MinxEnt, which is based on the marginal distributions of $g(\mathbf{x})$. The modified version is called the *parametric MinxEnt* (PME) method. We focus on the estimation of the rare-event probability

$$\ell = \mathbb{E}_{\mathbf{u}}\left[I_{\{S(\mathbf{X})\geq\gamma\}}\right],$$

where we assume for simplicity that $\mathbf{X} = (X_1, \ldots, X_n)$ is a binary random vector with independent components with probabilities $\mathbf{u} = (u_1, \ldots, u_n)$, that is, $\mathbf{X} \sim \mathsf{Ber}(\mathbf{u})$. Let $f(\mathbf{x}; \mathbf{u})$ denote the corresponding discrete pdf. We can estimate ℓ via the likelihood ratio estimator as

$$\widehat{\ell} = \frac{1}{N}\sum_{k=1}^{N}\left[I_{\{S(\mathbf{X}_k)\geq\gamma\}}\frac{f(\mathbf{X}_k;\mathbf{u})}{f(\mathbf{X}_k;\mathbf{p})}\right], \tag{9.41}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from $\mathsf{Ber}(\mathbf{p})$, for some probability vector $\mathbf{p}$, typically different from $\mathbf{u}$. The question is how to obtain a $\mathbf{p}$ that gives a low variance for the estimator $\widehat{\ell}$. If ℓ is related to a *counting* problem, as in (9.8), the *same* $\mathbf{p}$ can be used in (9.5) to estimate $|\mathscr{X}^*|$.

Let $g(\mathbf{x})$ in (9.34) be the optimal MinxEnt solution for this estimation problem. By summing $g(\mathbf{x})$ over all $x_i, i \neq j$, we obtain the marginal pdf for the j-th component. In particular, let $g(\mathbf{x})$ be the MinxEnt pdf, as in (9.34), and $h(\mathbf{x}) = f(\mathbf{x}; \mathbf{u})$, the prior pdf; then under g we have $X_j \sim \mathsf{Ber}(p_j)$, with

$$p_j = \mathbb{E}_g[X_j] = \frac{\sum_{\mathbf{x}} x_j\, h(\mathbf{x})\, \mathrm{e}^{\lambda S(\mathbf{x})}}{\mathbb{E}_h[\mathrm{e}^{\lambda S(\mathbf{X})}]},$$

so that

$$p_j = \frac{\mathbb{E}_{\mathbf{u}}\left[X_j\, \mathrm{e}^{\lambda S(\mathbf{X})}\right]}{\mathbb{E}_{\mathbf{u}}\left[\mathrm{e}^{\lambda S(\mathbf{X})}\right]}, \quad j = 1, \ldots, n, \tag{9.42}$$

with λ satisfying (9.35). Note that these $\{p_j\}$ will form our importance sampling parameter vector $\mathbf{p}$. Observe also that (9.42) was extensively used in [25] for updating the parameter vector $\mathbf{p}$ in rare-event estimation and for combinatorial optimization problems. Finally, it is crucial to recognize that (9.42) is similar to the corresponding CE formula (see also (5.67))

$$p_j = \frac{\mathbb{E}_{\mathbf{u}}[X_j\, I_{\{S(\mathbf{X})\geqslant\gamma\}}]}{\mathbb{E}_{\mathbf{u}}[I_{\{S(\mathbf{X})\geqslant\gamma\}}]}, \tag{9.43}$$

with one main difference: the indicator function $I_{\{S(\mathbf{X})\geqslant\gamma\}}$ in the CE formula is replaced by $\exp\{\lambda\, S(\mathbf{X})\}$. We shall call p_j in (9.42) the *optimal PME parameter* and consider it as an alternative to (9.43).

Remark 9.5.3 (PME for Exponential Families) The PME updating formula (9.42) can be generalized to hold for any exponential family parameterized by the mean in the same way that the CE updating formula (9.43) holds for such families. More specifically, suppose that under prior $h(\mathbf{x}) = f(\mathbf{x}; \mathbf{u})$ the random vector $\mathbf{X} = (X_1, \ldots, X_n)$ has independent components, and that each X_i is distributed according to some one-parameter exponential family $f_i(x_i; u_i)$ that is parameterized by its mean — thus, $\mathbb{E}_h[X_i] = \mathbb{E}_{\mathbf{u}}[X_i] = u_i$, with

$\mathbf{u} = (u_1, \ldots, u_n)$. The expectation of X_j under the MinxEnt solution is (in the continuous case)

$$\mathbb{E}_g[X_j] = \frac{\int x_j \, h(\mathbf{x}) \, \mathrm{e}^{\lambda S(\mathbf{x})} \, d\mathbf{x}}{\mathbb{E}_h[\mathrm{e}^{\lambda S(\mathbf{X})}]} = \frac{\mathbb{E}_{\mathbf{u}}\left[X_j \, \mathrm{e}^{\lambda S(\mathbf{X})}\right]}{\mathbb{E}_{\mathbf{u}}\left[\mathrm{e}^{\lambda S(\mathbf{X})}\right]}, \quad j = 1, \ldots, n \,. \tag{9.44}$$

Let $\mathbf{v} = (v_1, \ldots, v_n)$ be another parameter vector for the exponential family. Then the above analysis suggests carrying out importance sampling using v_j equal to $\mathbb{E}_g[X_j]$ given in (9.44).

Another way of looking at this is that $\mathbf{v}$ is chosen such that the Kullback-Leibler distance from the Boltzmann-like distribution $b(\mathbf{x}) \propto f(\mathbf{x}; \mathbf{u}) \, \mathrm{e}^{\lambda S(\mathbf{x})}$ to $f(\mathbf{x}; \mathbf{v})$ is minimized. Namely, minimizing $\mathcal{D}(b, f(\cdot; \mathbf{v}))$ with respect to $\mathbf{v}$ is equivalent to maximizing

$$\int h(\mathbf{x}) \, \mathrm{e}^{\lambda S(\mathbf{x})} \ln f(\mathbf{x}; \mathbf{v}) \, d\mathbf{x} = \mathbb{E}_{\mathbf{u}}[\mathrm{e}^{\lambda S(\mathbf{x})} \ln f(\mathbf{X}; \mathbf{v})] \,,$$

which (see (A.15) in Section A.3 of the Appendix) leads directly to the updating formula (9.44). Compare this with the standard CE method, where the Kullback-Leibler distance from $g^*(\mathbf{x}) \propto f(\mathbf{x}; \mathbf{u}) I_{\{S(\mathbf{x}) \geqslant \gamma\}}$ to $f(\mathbf{x}; \mathbf{v})$ is minimized instead.

Note that

1. For a separable function $S(\mathbf{x})$ MinxEnt *reduces* to PME. Recall that in this case the optimal joint pdf presents a product of marginal pdfs. Thus, PME is well suited for separable functions, such as those occurring in SATs (see (9.7)). However, it follows from Remark 9.5.2 that, even for separable functions, PME is *essentially different* from ECM.

2. Similar to CE, the optimal PME updating p_i^* and its estimators can be obtained analytically and on-line.

3. One does not need to resort to the MinxEnt program and to its joint pdf in order to derive the optimal parameters p_i^*.

4. The optimal λ^* is the same in both MinxEnt and PME.

5. Sampling from the marginal discrete pdf with the optimal parameters $\{p_i^*\}$ is easy and is similar to CE.

PME is well suited for separable functions (see item 1 above) and, it will turn out, also for block-separable functions, such as those that occur in the SAT problem (see (9.7)). Here *block-separable* means a function of the form

$$S(\mathbf{x}) = S_1(\mathbf{y}_1) + \cdots + S_m(\mathbf{y}_m) \,,$$

where each vector $\mathbf{y}_i$ depends on at most $r < n$ variables in $\{x_1, \ldots, x_n\}$.

One might wonder why the PME parameter in (9.42) would be preferable to the standard CE one in (9.43). The answer lies in the fact that in complex simulation-based models the PME optimal parameters $\mathbf{p}$ and λ are typically not available analytically and need to be estimated via Monte Carlo simulation. It turns out — this is discussed below — that for separable and block-separable function the corresponding estimator of (9.42) can have a significantly lower variance than the estimator of (9.43). This, in turn, means that the

variance of the estimator $\widehat{\ell}$, and for a counting problem the variance of estimator $\widehat{|\mathscr{X}^*|}$, will be significantly reduced.

For the estimation of the PME optimal parameters $\mathbf{p}$ and λ one can use, as in the CE and MinxEnt methods, a dynamic (multilevel) approach in which the estimates are determined iteratively. This leads to the following updating formula for p_j at the t-th iteration:

$$\widehat{p}_{t,j} = \frac{\sum_{k=1}^{N} X_{kj} \exp\{\widehat{\lambda}_t S(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}{\sum_{k=1}^{N} \exp\{\widehat{\lambda}_t S(\mathbf{X}_k)\} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}, \tag{9.45}$$

where $\widehat{\lambda}_t$ is obtained from the solution of (9.40) and W denotes, as usual, the likelihood ratio.

Note that $-1/\lambda_t$ can be viewed as a temperature parameter. In contrast to simulated annealing, the temperature is chosen here optimally in the CE sense rather than heuristically. Also, in contrast to CE, where only the elite sample is used while updating $\mathbf{p}$, in PME (see (9.45)) the entire sample is used.

We explain via a number of examples why the PME updating formula (9.45) can be much more stable than its CE counterpart,

$$\widehat{p}_{t,j} = \frac{\sum_{k=1}^{N} X_{kj} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}{\sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} W(\mathbf{X}_k; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}.$$

The key difference is that the number of product terms in W for CE is *always* n (the size of the problem):

$$W(\mathbf{X}; \mathbf{u}, \mathbf{p}_{t-1}) = \prod_{j=1}^{n} W_j(X_j; u_j, p_{t-1,j}), \quad W_j(X_j; u_j, p_{t-1,j}) = \frac{f_j(X_j; u_j)}{f_j(X_j; p_{t-1,j})}$$

irrespective of the form of $S(\mathbf{x})$, whereas the PME estimator (9.45) for separable or block-separable functions can be readily modified such that the number of product terms of the likelihood ratio is much smaller than n.

■ **EXAMPLE 9.8**

Consider a separable sample function, that is:

$$S(\mathbf{x}) = \sum_{j=1}^{n} S_j(x_j).$$

Then (9.42) reduces to

$$p_j = \frac{\mathbb{E}_{u_j}[X_j \exp\{\lambda S_j(X_j)\}]}{\mathbb{E}_{u_j}[\exp\{\lambda S_j(X_j)\}]}, \quad j = 1, \ldots, n,$$

which can be estimated via importance sampling as

$$\widehat{p}_{t,j} = \frac{\sum_{k=1}^{N} X_{kj} \exp\{\widehat{\lambda}_t S_j(X_{kj})\} W_j(X_{kj}; u_j, \widehat{p}_{t-1,j})}{\sum_{k=1}^{N} \exp\{\widehat{\lambda}_t S_j(X_{kj})\} W_j(X_{kj}; u_j, \widehat{p}_{t-1,j})} . \tag{9.46}$$

While both (9.45) and (9.46) represent unbiased estimators of p_j, the former involves the highly unstable likelihood ratio $W(\mathbf{X}) = \prod_{i=1}^{N} W_j(X_j)$ with respect to the n-dimensional vector $\mathbf{X}$, whereas the latter involves only the likelihood ratio with respect to the one-dimensional variable X_j. As a consequence, the estimator in (9.46) generally will have a much smaller variance than the one in (9.45).

■ **EXAMPLE 9.9**

Consider next a block-separable function $S(\mathbf{x})$ of the form

$$S(\mathbf{x}) = \sum_{j=1}^{n-1} S_j(x_j, x_{j+1}) .$$

Suppose we want to estimate p_2. Consider first the case where $S_3 \equiv 0$. Define $J = \{1, 2, 3\}$ and $\bar{J} = \{4, \ldots, n\}$. Let us denote by $\mathbf{x}_J$ the vector with components $\{x_j, j \in J\}$ and similar for $\mathbf{x}_{\bar{J}}$. We can now write $S(\mathbf{x})$ as

$$S(\mathbf{x}) = S_J(\mathbf{x}_J) + S_{\bar{J}}(\mathbf{x}_{\bar{J}}) , \tag{9.47}$$

with $S_J(\mathbf{X}_J) = S_1(X_1, X_2) + S_2(X_2, X_3)$ and $S_{\bar{J}}(\mathbf{X}_{\bar{J}}) = \sum_{j=3}^{n-1} S_j(X_j, X_{j+1})$ being *independent*. In this case, according to (9.42), the component p_j, $j = 2$ of $\mathbf{p}$ can be updated according to

$$\begin{aligned} p_j &= \frac{\mathbb{E}_{\mathbf{u}}\left[X_j \exp\left\{\lambda\left(S_J(\mathbf{X}_J) + S_{\bar{J}}(\mathbf{X}_{\bar{J}})\right)\right\}\right]}{\mathbb{E}_{\mathbf{u}}\left[\exp\left\{\lambda\left(S_J(\mathbf{X}_J) + S_{\bar{J}}(\mathbf{X}_{\bar{J}})\right)\right\}\right]} \\ &= \frac{\mathbb{E}_{\mathbf{u}_J}\left[X_j \exp\left\{\lambda S_J(\mathbf{X}_J)\right\}\right]}{\mathbb{E}_{\mathbf{u}_J}\left[\exp\left\{\lambda S_J(\mathbf{X}_J)\right\}\right]} , \end{aligned} \tag{9.48}$$

which can be estimated via importance sampling as

$$\widehat{p}_{t,j} = \frac{\sum_{k=1}^{N} X_{kj} \exp\{\widehat{\lambda}_t S_J(\mathbf{X}_{kJ})\} W_J(\mathbf{X}_{kJ}; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})}{\sum_{k=1}^{N} \exp\{\widehat{\lambda}_t S(\mathbf{X}_{kJ})\} W_J(\mathbf{X}_{kJ}; \mathbf{u}, \widehat{\mathbf{p}}_{t-1})} , \tag{9.49}$$

with W_J the corresponding likelihood ratio, which now only has two product terms. Thus, for large n, the estimator in (9.49) typically has a much smaller variance than the one in (9.45)

Suppose next that S_3 does not vanish, so that $S_J(\mathbf{X}_J)$ and $S_{\bar{J}}(\mathbf{X}_{\bar{J}})$ are dependent. Then, obviously, (9.45) is no longer valid. Nevertheless, for block-separable functions, this formula can still be used as an approximation to the true updating formula (9.42). In this case the estimator based on (9.45) may contain some bias, but

the advantage of estimating p_j with a greatly reduced variance could outweigh the disadvantage of having a slightly biased estimator.

We call (9.49) the *PME estimator*. Our main application of block-separable functions is the SAT counting problem, as explained in the next example.

■ **EXAMPLE 9.10**

Consider a 3-SAT problem defined by the following 8×6 clause matrix $A = (a_{ij})$:

$$A = \begin{pmatrix} 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \end{pmatrix}.$$

Let $S(\mathbf{x}) = \sum_{i=1}^{m} C_i(\mathbf{x})$ denote the number of valid clauses for truth assignment $\mathbf{x}$; see (9.7). Recall that $C_i(\mathbf{x})$ is the truth value (0 or 1) corresponding to the i-th clause and is given by (9.1). The block-separable structure of $S(\mathbf{x})$ is evident. Hence, we could use the PME estimator (9.49) to update p_j for each $j = 1, \ldots, n$. To indicate that the corresponding index set J depends on j, we denote it by J_j. The set J_j is constructed as follows. First, let K_j be the set of indices k for which a_{kj} is nonzero. For example, $K_2 = \{1, 4, 5\}$. Now define J_j as the set $\{r : a_{kr} \neq 0 \text{ for some } k \in K_j\}$. For example, $J_2 = \{2, 4, 6, 8\}$, which corresponds to $S_{J_2}(X_2, X_4, X_6, X_8) = C_1(X_2, X_4, X_6) + C_4(X_2, X_4, X_8) + C_5(X_2, X_4, X_6)$, where we have used a slight abuse of notation (note that $C_i(\mathbf{x})$ is formally a function of the eight-dimensional vector $\mathbf{x}$). The sets $J_j, j = 1, \ldots, 8$ are summarized in Table 9.4.

Table 9.4 The index sets $J_j, j = 1, \ldots, 8$ corresponding to clause matrix A.

j	J_j	j	J_j
1	1,2,3,5,7,8	5	1,3,5,8
2	2,4,6,8	6	2,4,6
3	1,3,5	7	1,4,7
4	1,2,4,6,7,8	8	1,2,4,5,8

Remark 9.5.4 (Depth-k Updating) Consider again the block-separable function $S(\mathbf{x}) = S_1(x_1, x_2) + S_2(x_2, x_3) + \cdots + S_{n-1}(x_{n-1}, x_n)$ in Example 9.9. To update p_j via the PME estimator (9.49), we need to identify the index set $J_j = \{k : x_k \text{ is in the same block as } x_j\}$. For example, $J_2 = \{1, 2, 3\}$ and $J_3 = \{2, 3, 4\}$. Let $J_2^{(2)} = \cup_{k \in J_2} J_k$ be the set of indices that are in the same block as at least one of the elements in J_2. Thus, in this example $J_2^{(2)} = \{1, 2, 3, 4\}$. Instead of updating p_2 via $J = J_2$ in (9.49), one could take $J = J_2^{(2)}$ instead. We call this *depth-2 updating*. By similar reasoning, one can define depth-3, depth-4, and so on, updating. For example, the depth-4 index set for p_2 is $J_2^{(4)} = \{1, 2, 3, 4, 5, 6\}$. As a final example, the depth-2 index set for p_3 in the 3-SAT problem of Example 9.10 is $J_3^{(2)} = J_1 \cup J_3 \cup J_5 = \{1, 2, 3, 5, 7, 8\}$

Our extensive numerical results indicate that although the depth-1 solution typically provides satisfactory results, depth-2 performs best in terms of small bias and high accuracy. It is interesting to note that in contrast to CE, while using the naive PME updating for $\mathbf{p}$ (with the size of W equal to the size n of the problem), we still obtain reasonably good results. This is quite a surprising result, which needs further investigations.

We saw that the PME estimator (9.49) is ideally suited for componentwise separable functions. Let us examine its efficiency (in terms of the size of the index set J) for the K-RSAT problem, having in mind that $m \approx \beta^* n$, where β^* is the threshold value. Consider first the 2-RSAT problem, where $\beta^* = 1$. Each variable x_j or its negation $\bar{x}_j$ occurs on average in two different clauses. Since each of these classes has a 1 or -1 in column j and also in another column (possibly the same), this implies that the average size of the index set J is less than 3 — close to 3 for large n. A similar analysis shows that for a K-RSAT problem the average size of J each is less than $\beta K(K-1)+1$, provided that this is less than n. For 3-SAT, for example, with the threshold $\beta = 4.26$, we have $4.26 \times 6 + 1 \approx 26$, provided that $n > 26$. Note that for large K, the PME estimator could still be quite noisy due to the size of each W term.

As mentioned, the PME method for rare events and counting is similar to CE in the sense that the updating rule in (9.49) is parametric. The main difference is that the updating rule in CE involves a likelihood ratio W that is of size n, while the updating rule for PME is based on block-separable functions of type (9.49) with the size of W much smaller than n.

Below we present the PME algorithm for counting SAT assignments based on the updating formula (9.49).

Algorithm 9.5.1 (PME Algorithm for Counting SAT Assignments)

1. *Define* $\widehat{\mathbf{p}}_0 = \mathbf{u}$. *Set* $t = 1$ *(iteration = level counter).*
2. *Generate a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *from the density* $f(\mathbf{x}; \widehat{\mathbf{p}}_{t-1})$ *and compute* $\widehat{\gamma}_t$ *according to* (9.39).
3. *Use the* **same** *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *and update* $\mathbf{p}_t$ *according to* (9.49). *Denote the solution by* $\widehat{\mathbf{p}}_t$.
4. *Smooth out the vector* $\widehat{\mathbf{p}}_t$, *similar to* (8.20).
5. *If* $\widehat{\gamma}_t < m$, *set* $t = t + 1$ *and reiterate from Step 2. Otherwise, proceed with Step 6.*
6. *Estimate the counting quantity* $|\mathscr{X}^*|$ *as*

$$\widehat{|\mathscr{X}^*|} = \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant m\}} \frac{1}{f(\mathbf{X}_k; \widehat{\mathbf{p}}_T)}, \tag{9.50}$$

where T *denotes the final number of iterations (= number of levels used).*

9.6 PME FOR COMBINATORIAL OPTIMIZATION PROBLEMS AND DECISION MAKING

It should follow from the above that the PME counting Algorithm 9.5.1 can be readily modified to handle difficult decision-making problems (like deciding whether or not there exists a valid SAT assignment) and combinatorial optimization problems. In particular, below we design a PME algorithm for combinatorial optimization problems, which coincides with its counting (rare-event) PME counterpart Algorithm 9.5.1, provided that its likelihood ratios are removed. To be more specific, in the combinatorial optimization algorithm we will be using Steps 1–4 of Algorithm 9.5.1 with likelihood ratios set automatically to unity. Recall that the standard CE algorithm for combinatorial optimization problems coincides with its CE counting (rare-events) counterpart by setting all the relevant likelihood ratios to unity or simply by removing them.

For completeness, we present below the PME algorithm for combinatorial optimization problems with Bernoulli updating formulas. Its extension to the case of n-point discrete pdf is similar.

Algorithm 9.6.1 (PME Algorithm for Combinatorial Optimization Problems)

1. *Define* $\widehat{\mathbf{p}}_0 = \mathbf{u}$. *Set* $t = 1$ *(iteration = level counter).*
2. *Generate a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *from the density* $f(\mathbf{x}; \widehat{\mathbf{p}}_{t-1})$ *and compute* $\widehat{\gamma}_t$ *according to* (9.39).
3. *Use the* **same** *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *and update* $\mathbf{p}_t$ *according to* (9.49) *by setting all likelihood ratios* $W = 1$. *Denote the solution by* $\widehat{\mathbf{p}}_t$.
4. *Smooth out the vector* $\widehat{\mathbf{p}}_t$ *similar to* (8.20).
5. *If the stopping criterion is met, stop; otherwise, set* $t = t + 1$ *and return to Step 2.*

Recall from Algorithm 8.3.1 that as a stopping criterion one can use (8.21); that is, if for some $t \geq d$, say $d = 5$,

$$\widehat{\gamma}_t = \widehat{\gamma}_{t-1} = \cdots = \widehat{\gamma}_{t-d} ,$$

then stop.

Our extensive numerical studies with Algorithm 9.6.1 (see also [25]) suggest that it is at least as accurate as its CE counterpart, Algorithm 8.3.1. We argue that the main reasons are that

1. The PME Algorithm 9.6.1 uses the entire sample instead of only the elite one.
2. The temperature parameter $-1/\lambda_t$ is chosen optimally in the MinxEnt sense rather than heuristically.

9.7 NUMERICAL RESULTS

Here we present comparative simulation studies with the CE and PME algorithms for different K-RSAT problems. All of our simulation results correspond to the *difficult rare-event* cases, in the sense that $\beta = m/n$ is chosen near the critical value β^*. For 2-RSAT and 3-RSAT, $\beta^* = 1$ and $\beta^* \approx 4.2$, respectively.

If not stated otherwise, we set $\varrho = 0.001$ and $\alpha = 0.7$, and we use equal sample sizes N for each run of PME and CE while estimating both $\mathbf{p}$ and $|\mathscr{X}^*|$.

To study the variability in the solutions, we run each problem 10 times and report our statistics based on these 10 runs. In the following tables t denotes the iteration number. The other quantities are defined as follows (for each iteration t):

1. *Mean, max and min* $\widehat{|\mathscr{X}^*|}$ denote the sample mean, maximum, and minimum of the 10 estimates of $|\mathscr{X}^*|$, respectively.
2. *Mean, max and min Found* denote the sample mean, maximum, and minimum of the number of different valid assignments found in each of the 10 samples of size N, respectively. Note that the maximum value can be viewed as the lower bound of the true unknown quantity $|\mathscr{X}^*|$.
3. PV denotes the proportion of generated valid assignments averaged over 10 replications.
4. RE denotes the mean relative error for $\widehat{|\mathscr{X}^*|}$ averaged over the 10 runs.

We consider the following models.

1. Random 3-SAT with a relatively small matrix, namely $A = (25 \times 100)$. For this matrix we found the exact number of true SAT assignments via full enumeration. We then compared this exact number with the one generated by the CE and PME methods. We found that PME performs very accurately, delivering after 10 iterations a relative error of less than 0.5% and outperforming CE.
2. Random 3-SAT with a relatively large matrix $A = (75 \times 375)$, which is taken from the SATLIB website `www.satlib.org`. We found again that PME performs quite accurately, while CE fails.

We first consider the random 3-SAT problem for $n = 25$ with $\beta^* \approx 4.2$, that is, near the critical value. For this case, six true assignments were found via full enumeration. Tables 9.5 and 9.6 present the performance of the PME and CE algorithms, respectively, for the random 3-SAT with matrix $A = (25 \times 100)$ and sample size $N = 50{,}000$. It took about 5 seconds to compute each table.

Table 9.5 Performance of the PME algorithm for the random 3-SAT with the clause matrix $A = (25 \times 100)$, six valid assignments, and $N = 50{,}000$.

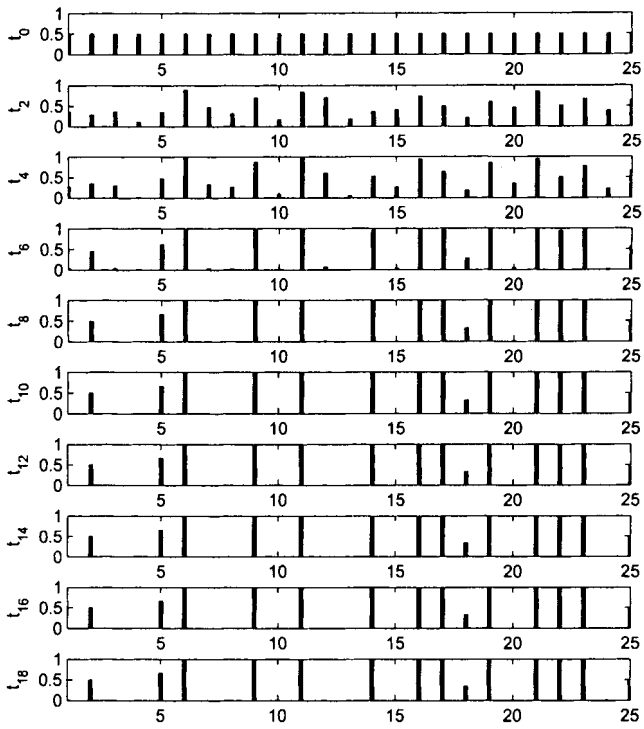
	$\widehat{\lvert\mathscr{X}^*\rvert}$			Found				
t	Mean	Max	Min	Mean	Max	Min	PV	RE
1	6.1001	35.4294	0.0000	0.2000	1	0	0.0000	2.0324
2	5.4844	11.9808	0.0000	2.0000	5	0	0.0001	0.6773
3	5.1513	8.5898	3.2131	4.9000	6	3	0.0009	0.3003
4	5.6476	6.9381	3.5681	5.7000	6	5	0.0429	0.1477
5	5.9956	6.1841	5.7894	6.0000	6	6	0.1863	0.0166
6	6.0130	6.0792	5.9370	6.0000	6	6	0.3362	0.0072
7	6.0077	6.0589	5.9203	6.0000	6	6	0.4082	0.0068
8	5.9975	6.0511	5.9655	6.0000	6	6	0.4325	0.0046
9	6.0057	6.0398	5.9637	6.0000	6	6	0.4418	0.0039
10	6.0075	6.0446	5.9445	6.0000	6	6	0.4439	0.0046

Table 9.6 Performance of the CE algorithm for the random 3-SAT with the clause matrix $A = (25 \times 100)$, six valid assignments, and $N = 50{,}000$.

	$\widehat{\lvert\mathscr{X}^*\rvert}$			Found				
t	Mean	Max	Min	Mean	Max	Min	PV	RE
1	7.0180	49.5243	0.0000	0.2000	1	0	0.0000	2.2014
2	7.9630	27.3171	0.0000	0.9000	3	0	0.0000	1.2339
3	6.2836	10.8166	0.0000	3.7000	6	0	0.0003	0.4660
4	10.4858	31.4573	0.0000	1.1000	3	0	0.0000	0.9220
5	5.6237	8.3948	2.3175	2.9000	6	2	0.0001	0.3110
6	5.5700	7.9109	1.0197	4.0000	6	2	0.0004	0.3457
7	5.5330	10.6799	1.6970	4.2000	6	2	0.0112	0.4508
8	5.2768	9.3622	1.7971	4.4000	6	3	0.0397	0.3854
9	5.2990	7.0927	2.8527	4.7000	6	3	0.0795	0.2475
10	4.9066	6.2015	2.3662	5.0000	6	3	0.1430	0.3276

Figure 9.9 presents a typical dynamics of the PME algorithm for the random 3-SAT problem with the clause matrix $A = (25 \times 100)$.

Figure 9.9 Typical dynamics of the PME algorithm for the random 3-SAT with the clause matrix $A = (25 \times 100)$, six valid assignments, and $N = 50{,}000$.



It readily follows from these data that after several iterations both methods are capable of finding all six true assignments, while the PME method is more accurate (compare, for example, RE $= 0.044$ for PME with RE $= 0.4185$ for CE at iteration $t = 18$). One can also see that most of the parameters p_i, $i = 1, \ldots, 25$ converge to the degenerate case, that is, they are equal to either 1 or to 0. This is due to the fact that the number of true assignments is small (equal to 6). It should be clear that if the number of true assignments equals 1, then *all* parameters p_i, $i = 1, \ldots, n$ would converge to the degenerate case.

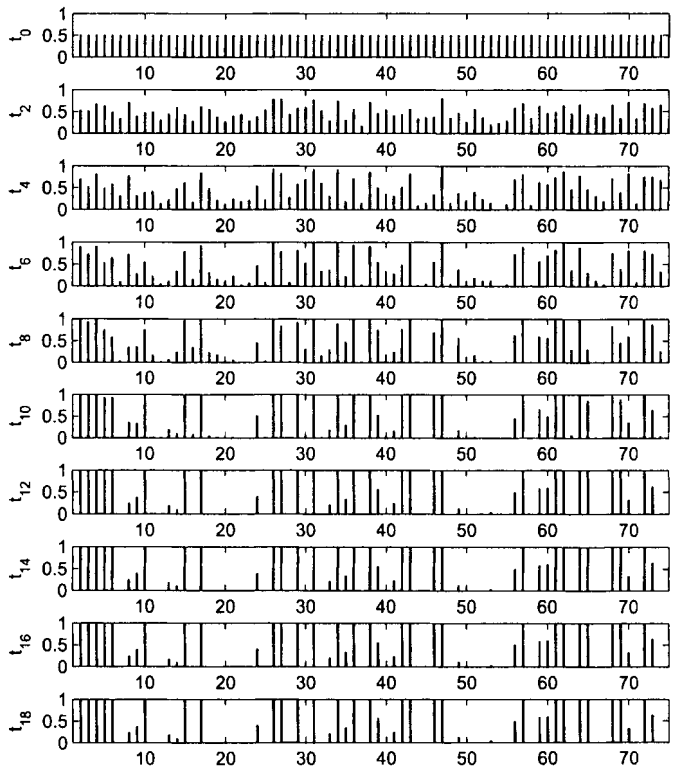
We next present, in Table 9.5 and Figure 9.10, the performance of the PME algorithm for the random 3-SAT with the matrix $A = (75 \times 325)$, taken from the SATLIB website `www.satlib.org`. We found again that in this case CE fails, in the sense that its relative error is about 0.6, while for PME the relative error is approximately 0.02. We also found that for $n > 50$ CE fails, while PME still performs nicely up to $n = 500$.

Note that block separability does not automatically guarantee that the size of the likelihood ratio term W in PME will always be small for all randomly generated matrices A. Note, however, that by limiting the number of literals in the matrix $A = (n \times \beta n)$ *columnwise*, as in the Lovasz local lemma [22], we can *always* guarantee the superiority of PME compared to CE, at least for moderate values of K, say $K \leq 5$, in the random K-SAT models.

Table 9.7 Performance of the PME algorithm for the random 3-SAT with the clause matrix $A = (75 \times 325)$ and $N = 100{,}000$.

	$\widehat{\lvert\mathscr{X}^*\rvert}$			Found				
t	Mean	Max	Min	Mean	Max	Min	PV	RE
6	0.00	0.00	0.00	0.00	0	0	0.0000	NaN
7	382.08	1818.15	0.00	4.70	35	0	0.0000	1.7765
8	1349.59	3152.26	0.00	110.30	373	0	0.0018	0.8089
9	1397.32	2767.18	525.40	467.70	1089	42	0.0369	0.4356
10	1375.68	1828.11	878.00	859.50	1237	231	0.1143	0.1755
11	1434.95	1776.47	1341.54	1153.70	1268	910	0.2020	0.0880
12	1374.64	1423.99	1340.12	1244.90	1284	1180	0.2529	0.0195
13	1392.17	1441.19	1356.97	1273.10	1290	1248	0.2770	0.0207
14	1397.13	1466.46	1358.02	1277.30	1291	1260	0.2816	0.0250
15	1384.37	1419.97	1354.32	1277.10	1296	1258	0.2832	0.0166
16	1377.75	1424.07	1320.23	1271.90	1284	1251	0.2870	0.0258

Figure 9.10 Typical dynamics of the PME algorithm for the random 3-SAT problem with the clause matrix $A = (75 \times 325)$ and $N = 100{,}000$.



PROBLEMS

9.1 Prove the upper bound (9.21).

9.2 Prove the upper bound (9.22).

9.3 Consider Problem 8.9. Implement and run a PME algorithm on this synthetic max-cut problem for a network with $n = 400$ nodes, with $m = 200$. Compare with the CE algorithm.

9.4 Let $\{A_i\}$ be an arbitrary collection of subsets of some finite set $\mathscr{X}$. Show that

$$|\cup_i A_i| = \sum |A_i| - \sum_{i<j} |A_i \cap A_j| + \sum_{i<j<k} |A_i \cap A_j \cap A_k| - \cdots .$$

This is the useful *inclusion-exclusion* principle.

9.5 A famous problem in combinatorics is the *distinct representatives* problem, which is formulated as follows. Given a set $\mathscr{A}$ and subsets $\mathscr{A}_1, \ldots, \mathscr{A}_n$ of $\mathscr{A}$, is there a vector $\mathbf{x} = (x_1, \ldots, x_n)$ such that $x_i \in \mathscr{A}_i$ for each $i = 1, \ldots, n$ and the $\{x_i\}$ are all distinct (that is, $x_i \neq x_j$ if $i \neq j$)?

a) Suppose, for example, that $\mathscr{A} = \{1,2,3,4,5\}$, $\mathscr{A}_1 = \{1,2,5\}$, $\mathscr{A}_2 = \{1,4\}$, $\mathscr{A}_3 = \{3,5\}$, $\mathscr{A}_4 = \{3,4\}$, and $\mathscr{A}_5 = \{1\}$. Count the total number of distinct representatives.

b) Argue why the total number of distinct representatives in the above problem is equal to the *permanent* of the following matrix A.

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} .$$

9.6 Let $X_1, \ldots, X_n$ be independent random variables, each with marginal pdf f. Suppose we wish to estimate $\ell = \mathbb{P}_f(X_1 + \cdots + X_n \geqslant \gamma)$ using MinxEnt. For the prior pdf, one could choose $h(\mathbf{x}) = f(x_1)f(x_2)\cdots f(x_n)$, that is, the joint pdf. We consider only a single constraint in the MinxEnt program, namely, $S(\mathbf{x}) = x_1 + \cdots + x_n$. As in (9.34), the solution to this program is given by

$$g(\mathbf{x}) = c\, h(\mathbf{x})\, \mathrm{e}^{\lambda S(\mathbf{x})} = c \prod_{j=1}^{n} \mathrm{e}^{\lambda x_j} f(x_j) ,$$

where $c = 1/\mathbb{E}_h[\mathrm{e}^{\lambda S(\mathbf{X})}] = (\mathbb{E}_f[\mathrm{e}^{\lambda X}])^{-n}$ is a normalization constant and λ satisfies (9.35). Show that the new marginal pdfs are obtained from the old ones by an *exponential twist*, with twisting parameter $-\lambda$; see also (A.13).

9.7 Problem 9.6 can be generalized to the case where $S(\mathbf{x})$ is a coordinatewise separable function, as in (9.36), and the components $\{X_i\}$ are independent under the prior pdf $h(\mathbf{x})$. Show that also in this case the components under the optimal MinxEnt pdf $g(\mathbf{x})$ are independent and determine the marginal pdfs.

9.8 Let $\mathscr{X}$ be the set of permutations $\mathbf{x} = (x_1, \ldots, x_n)$ of the numbers $1, \ldots, n$, and let

$$S(\mathbf{x}) = \sum_{j=1}^{n} j\, x_j \,. \tag{9.51}$$

Let $\mathscr{X}^* = \{\mathbf{x} : S(\mathbf{x}) \geq \gamma\}$, where γ is chosen such that $|\mathscr{X}^*|$ is very small relative to $|\mathscr{X}| = n!$.

Implement a randomized algorithm to estimate $|\mathscr{X}^*|$ based on (9.9), using $\mathscr{X}_j = \{\mathbf{x} : S(\mathbf{x}) \geq \gamma_j\}$, for some sequence of $\{\gamma_j\}$ with $0 = \gamma_0 < \gamma_1 < \cdots < \gamma_r = \gamma$. Estimate each quantity $\mathbb{P}_U(\mathbf{X} \in \mathscr{X}_k \mid \mathbf{X} \in \mathscr{X}_{k-1})$, using the Metropolis–Hastings algorithm for drawing from the uniform distribution on $\mathscr{X}_{k-1}$. Define two permutations $\mathbf{x}$ and $\mathbf{y}$ as neighbors if one can be obtained from the other by swapping two indices, for example $(1, 2, 3, 4, 5)$ and $(2, 1, 3, 4, 5)$.

9.9 Write the Lagrangian dual problem for the MinxEnt problem with constraints in Remark 9.5.1.

Further Reading

For good references on #P-complete problems with emphasis on SAT problems see [21, 22]. The counting class #P was defined by Valiant [29]. The FPRAS for counting SATs in DNF is due to Karp and Luby [18], who also give the definition of FPRAS. The first FPRAS for counting the volume of a convex body was given by Dyer et al. [10]. See also [8] for a general introduction to random and nonrandom algorithms. Randomized algorithms for approximating the solutions of some well-known counting #P-complete problems and their relation to MCMC are treated in [11, 14, 22, 23, 28]. Bayati and Saberi [1] propose an efficient importance sampling algorithm for generating graphs uniformly at random. Chen et al. [7] discuss the efficient estimation, via sequential importance sampling, of the number of 0-1 tables with fixed row and column sums. Blanchet [4] provides the first importance sampling estimator with bounded relative error for this problem. Roberts and Kroese [24] count the number of paths in arbitrary graphs using importance sampling.

Since the pioneering work of Shannon [27] and Kullback [19], the relationship between statistics and information theory has become a fertile area of research. The work of Kapur and Kesavan, such as [16, 17], has provided great impetus to the study of entropic principles in statistics. Rubinstein [25] introduced the idea of updating the probability vector for combinatorial optimization problems and rare events using the marginals of the MinxEnt distribution. The above PME algorithms for counting and combinatorial optimization problems present straightforward modifications of the ones given in [26]. For some fundamental contributions to MinxEnt see [2, 3]. In [5, 6] a powerful generalization and unification of the ideas behind the MinxEnt and CE methods is presented under the name *generalized cross-entropy* (GCE).

REFERENCES

1. M. Bayati and A. Saberi. Fast generation of random graphs via sequential importance sampling. Manuscript, Stanford University, 2006.
2. A. Ben-Tal, D. E. Brown, and R. L. Smith. Relative entopy and the convergence of the posterior and empirical distributions under incomplete and conflicting information. Manuscript, University of Michigan, 1988.

3. A. Ben-Tal and M. Teboulle. Penalty functions and duality in stochastic programming via ϕ divergence functionals. *Mathematics of Operations Research*, 12:224–240, 1987.

4. J. Blanchet. Importance sampling and efficient counting of 0-1 contingency tables. In *Valuetools '06: Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools*, page 20. ACM Press, New York, 2006.

5. Z. I. Botev. *Stochastic Methods for Optimization and Machine Learning.* ePrintsUQ, http://eprint.uq.edu.au/archive/00003377/, BSc (Hons) Thesis, Department of Mathematics, School of Physical Sciences, The University of Queensland, 2005.

6. Z. I. Botev, D. P. Kroese, and T. Taimre. Generalized cross-entropy methods for rare-event simulation and optimization. *Simulation: Transactions of the Society for Modeling and Simulation International*, 2007. In press.

7. Y. Chen, P. Diaconis, S. P. Holmes, and J. Liu. Sequential Monte Carlo method for statistical analysis of tables. *Journal of the American Statistical Association*, 100:109–120, 2005.

8. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press and McGraw-Hill, 2nd edition, 2001.

9. T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, New York, 1991.

10. M. Dyer, A. Frieze, and R. Kannan. A random polynomial-time algorithm for approximation the volume of convex bodies. *Journal of the ACM*, 38:1–17, 1991.

11. C. P. Gomes and B. Selman. Satisfied with physics. *Science*, pages 784–785, 2002.

12. J. Gu, P. W. Purdom, J. Franco, and B. W. Wah. Algorithms for the satisfiability (SAT) problem: A survey. In *Satisfiability Problem: Theory and Applications*, volume 35 of DIMACS Series in Discrete Mathematics. American Mathematical Society, 1996.

13. E. T. Jaynes. *Probability Theory: The Logic of Science.* Cambridge University Press, Cambridge, 2003.

14. M. Jerrum. *Counting, Sampling and Integrating: Algorithms and Complexity.* Birkhauser Verlag, Basel, 2003.

15. M. Jerrum and A. Sinclair. *Approximation Algorithms for NP-hard Problems*, chapter : The Markov chain Monte Carlo Method: An approach to approximate counting and integration. PWS, 1996.

16. J. N. Kapur and H. K. Kesavan. The generalized maximum entropy principle. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:1042–1052, 1989.

17. J. N. Kapur and H. K. Kesavan. *Entropy Optimization Principles with Applications.* Academic Press, New York, 1992.

18. R. M. Karp and M. Luby. Monte Carlo algorithms for enumeration and reliability problems. In *Proceedings of the 24-th IEEE Annual Symposium on Foundations of Computer Science*, pages 56–64, Tucson, 1983.

19. S. Kullback. *Information Theory and Statistics.* John Wiley & Sons, New York, 1959.

20. J. S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer-Verlag, New York, 2001.

21. M. Mézard and Andrea Montanari. *Constraint Satisfaction Networks in Physics and Computation.* Oxford University Press, Oxford, 2006.

22. M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, Cambridge, 2005.

23. R. Motwani and R. Raghavan. *Randomized Algorithms.* Cambridge University Press, Cambridge, 1997.

24. B. Roberts and D. P. Kroese. Estimating the number of s–t paths in a graph. *Journal of Graph Algorithms an Applications*, 2007. In press.

25. R. Y. Rubinstein. The stochastic minimum cross-entropy method for combinatorial optimization and rare-event estimation. *Methodology and Computing in Applied Probability*, 7:5–50, 2005.

26. R. Y. Rubinstein. How many needles are in a haystack, or how to solve #p-complete counting problems fast. *Methodology and Computing in Applied Probability*, 8(1):5 – 51, 2006.

27. C. E. Shannon. The mathematical theory of communications. *Bell Systems Technical Journal*, 27:623–656, 1948.

28. R. Tempo, G. Calafiore, and F. Dabbene. *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer-Verlag, London, 2004.

29. L.G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.

30. D. J. A. Welsh. *Complexity: Knots, Colouring and Counting*. Cambridge University Press, Cambridge, 1993.

APPENDIX

A.1 CHOLESKY SQUARE ROOT METHOD

Let Σ be a covariance matrix. We wish to find a matrix B such that $\Sigma = BB^T$. The *Cholesky square root method* computes a lower triangular matrix B via a set of recursive equations as follows: From (1.23) we have

$$Z_1 = b_{11}X_1 + \mu_1 \,. \tag{A.1}$$

Therefore, $\mathrm{Var}(Z_1) = \sigma_{11} = b_{11}^2$ and $b_{11} = \sigma_{11}^{1/2}$. Proceeding with the second component of (1.23), we obtain

$$Z_2 = b_{21}X_1 + b_{22}X_2 + \mu_2 \tag{A.2}$$

and thus

$$\sigma_{22} = \mathrm{Var}(Z_2) = \mathrm{Var}(b_{21}X_1 + b_{22}X_2) = b_{21}^2 + b_{22}^2 \,. \tag{A.3}$$

Further, from (A.1) and (A.2),

$$\sigma_{12} = \mathbb{E}[(Z_1 - \mu_1)(Z_2 - \mu_2)] = \mathbb{E}[b_{11}X_1(b_{21}X_1 + b_{22}X_2)] = b_{11}b_{21} \,. \tag{A.4}$$

Hence, from (A.3) and (A.4) and the symmetry of Σ,

$$b_{21} = \frac{\sigma_{12}}{b_{11}} = \frac{\sigma_{12}}{\sigma_{11}^{1/2}} \tag{A.5}$$

$$b_{22} = \left(\sigma_{22} - \frac{\sigma_{21}^2}{\sigma_{11}}\right)^{1/2} \,. \tag{A.6}$$

Simulation and the Monte Carlo Method, Second Edition. By R.Y. Rubinstein and D. P. Kroese

Generally, the b_{ij} can be found from the recursive formula

$$b_{ij} = \frac{\sigma_{ij} - \sum_{k=1}^{j-1} b_{ik} b_{jk}}{\left(\sigma_{jj} - \sum_{k=1}^{j-1} b_{jk}^2\right)^{1/2}}, \tag{A.7}$$

where, by convention,

$$\sum_{k=1}^{0} b_{ik} b_{jk} = 0, \quad 1 \leqslant j \leqslant i \leqslant n .$$

A.2 EXACT SAMPLING FROM A CONDITIONAL BERNOULLI DISTRIBUTION

Suppose the vector $\mathbf{X} = (X_1, \ldots, X_n)$ has independent components, with $X_i \sim \mathrm{Ber}(p_i)$, $i = 1, \ldots, n$. It is not difficult to see (see Problem A.1) that the conditional distribution of $\mathbf{X}$ given $\sum_i X_i = k$ is given by

$$\mathbb{P}\left(X_1 = x_1, \ldots, X_n = x_n \,\middle|\, \sum_{i=1}^{n} X_i = k \right) = \frac{\prod_{i=1}^{n} w_i^{x_i}}{c}, \tag{A.8}$$

where c is a normalization constant and $w_i = p_i/(1 - p_i)$, $i = 1, \ldots, n$. Generating random variables from this distribution can be done via the so-called *drafting* procedure, described, for example, in [2]. The Matlab code below provides a procedure for calculating the normalization constant c and drawing from the conditional joint pdf above.

■ **EXAMPLE A.1**

Suppose $\mathbf{p} = (1/2, 1/3, 1/4, 1/5)$ and $k = 2$. Then $\mathbf{w} = (w_1, \ldots, w_4) = (1, 1/2, 1/3, 1/4)$. The first element of `Rgens(k,w)`, with `k` $= 2$ and `w` $=$ `w` is $35/24 \approx 1.45833$. This is the normalization constant c. Thus, for example,

$$\mathbb{P}\left(X_1 = 0, X_2 = 1, X_3 = 0, X_4 = x_1 \,\middle|\, \sum_{i=1}^{4} X_i = 2 \right) = \frac{w_2\, w_4}{35/24} = \frac{3}{35} \approx 0.08571 .$$

To generate random vectors according to this conditional Bernoulli distribution call `condbern(p,k)`, where `k` is the number of unities (here 2) and `p` is the probability vector `p`. This function returns the positions of the unities, such as (1, 2) or (2,4).

```
function sample = condbern(k,p)
% k = no of units in each sample, P = probability vector
W=zeros(1,length(p));
sample=zeros(1,k);
ind1=find(p==1);
sample(1:length(ind1))=ind1;
k=k-length(ind1);
ind=find(p<1 & p>0);
W(ind)=p(ind)./(1-p(ind));
for i=1:k
```

```
    Pr=zeros(1,length(ind));
    Rvals=Rgens(k-i+1,W(ind));
    for j=1:length(ind)
        Pr(j)=W(ind(j))*Rvals(j+1)/((k-i+1)*Rvals(1));
    end
    Pr=cumsum(Pr);
    entry=ind(min(find(Pr>rand)));
    ind=ind(find(ind~=entry));
    sample(length(ind1)+i)=entry;
end
sample=sort(sample);
return


function Rvals = Rgens(k,W)
N=length(W);
T=zeros(k,N+1);
R=zeros(k+1,N+1);
for i=1:k
    for j=1:N, T(i,1)=T(i,1)+W(j)^i; end
    for j=1:N, T(i,j+1)=T(i,1)-W(j)^i; end
end
R(1,:)=ones(1,N+1);
for j=1:k
    for l=1:N+1
        for i=1:j
            R(j+1,l)=R(j+1,l)+(-1)^(i+1)*T(i,l)*R(j-i+1,l);
        end
    end
    R(j+1,:)=R(j+1,:)/j;
end
Rvals=[R(k+1,1),R(k,2:N+1)];
return
```

A.3 EXPONENTIAL FAMILIES

Exponential families play an important role in statistics; see, for example, [1]. Let $\mathbf{X}$ be a random variable or vector (in this section, vectors will always be interpreted as *column* vectors) with pdf $f(\mathbf{x};\boldsymbol{\theta})$ (with respect to some measure), where $\boldsymbol{\theta} = (\theta_1,\ldots,\theta_m)^T$ is an m-dimensional parameter vector. $\mathbf{X}$ is said to belong to an m-parameter *exponential family* if there exist real-valued functions $t_i(\mathbf{x})$ and $h(\mathbf{x}) > 0$ and a (normalizing) function $c(\boldsymbol{\theta}) > 0$ such that

$$f(\mathbf{x};\boldsymbol{\theta}) = c(\boldsymbol{\theta})\, \mathrm{e}^{\boldsymbol{\theta}\cdot\mathbf{t}(\mathbf{x})}\, h(\mathbf{x})\,, \tag{A.9}$$

where $\mathbf{t}(\mathbf{x}) = (t_1(\mathbf{x}),\ldots,t_m(\mathbf{x}))^T$ and $\boldsymbol{\theta}\cdot\mathbf{t}(\mathbf{x})$ is the inner product $\sum_{i=1}^m \theta_i t_i(\mathbf{x})$. The representation of an exponential family is in general not unique.

Remark A.3.1 (Natural Exponential Family) The standard definition of an exponential family involves a family of densities $\{g(\mathbf{x};\mathbf{v})\}$ of the form

$$g(\mathbf{x};\mathbf{v}) = d(\mathbf{v})\,\mathrm{e}^{\boldsymbol{\theta}(\mathbf{v})\cdot\mathbf{t}(\mathbf{x})}\,h(\mathbf{x})\,, \tag{A.10}$$

where $\boldsymbol{\theta}(\mathbf{v}) = (\theta_1(\mathbf{v}),\ldots,\theta_m(\mathbf{v}))^T$, and the $\{\theta_i\}$ are real-valued functions of the parameter $\mathbf{v}$. By *reparameterization* — by using the θ_i as parameters — we can represent (A.10) in so-called *canonical form* (A.9). In effect, $\boldsymbol{\theta}$ is the natural parameter of the exponential family. For this reason, a family of the form (A.9) is called a *natural exponential family*.

Table A.1 displays the functions $c(\boldsymbol{\theta})$, $t_k(x)$, and $h(x)$ for several commonly used distributions (a dash means that the corresponding value is not used).

Table A.1 The functions $c(\boldsymbol{\theta})$, $t_k(x)$ and $h(x)$ for commonly used distributions.

Distr.	$t_1(x)$, $t_2(x)$	$c(\boldsymbol{\theta})$	θ_1, θ_2	$h(x)$
Gamma(α,λ)	x, $\ln x$	$\dfrac{(-\theta_1)^{\theta_2+1}}{\Gamma(\theta_2+1)}$	$-\lambda$, $\alpha-1$	1
N(μ,σ^2)	x, x^2	$\dfrac{\mathrm{e}^{\theta_1^2/(4\theta_2)}}{\sqrt{-\pi/\theta_2}}$	$\dfrac{\mu}{\sigma^2}$, $-\dfrac{1}{2\sigma^2}$	1
Weib(α,λ)	x^α, $\ln x$	$-\theta_1(\theta_2+1)$	$-\lambda^\alpha$, $\alpha-1$	1
Bin(n,p)	x, –	$(1+\mathrm{e}^{\theta_1})^{-n}$	$\ln\left(\dfrac{p}{1-p}\right)$, –	$\binom{n}{x}$
Poi(λ)	x, –	$\mathrm{e}^{-\mathrm{e}^{\theta_1}}$	$\ln\lambda$, –	$\dfrac{1}{x!}$
G(p)	$x-1$, –	$1-\mathrm{e}^{\theta_1}$	$\ln(1-p)$, –	1

As an important instance of a natural exponential family, consider the univariate, single-parameter ($m=1$) case with $t(x)=x$. Thus, we have a family of densities $\{f(x;\theta), \theta\in\Theta\subset\mathbb{R}\}$ given by

$$f(x;\theta) = c(\theta)\,\mathrm{e}^{\theta x}\,h(x)\,. \tag{A.11}$$

If $h(x)$ is a pdf, then $c^{-1}(\theta)$ is the corresponding *moment generating function*:

$$c^{-1}(\theta) = \int \mathrm{e}^{\theta x}\,h(x)\,dx\,.$$

It is sometimes convenient to introduce instead the logarithm of the moment generating function:

$$\zeta(\theta) = \ln\int \mathrm{e}^{\theta x}h(x)\,dx\,,$$

which is called the *cumulant function*. We can now write (A.11) in the following convenient form:

$$f(x;\theta) = \mathrm{e}^{\theta x-\zeta(\theta)}\,h(x)\,. \tag{A.12}$$

■ **EXAMPLE A.2**

If we take h as the density of the $\mathsf{N}(0,\sigma^2)$-distribution, $\theta = \lambda/\sigma^2$ and $\zeta(\theta) = \sigma^2\,\theta^2/2$, then the family $\{f(\cdot;\theta), \theta \in \mathbb{R}\}$ is the family of $\mathsf{N}(\lambda,\sigma^2)$ densities, where σ^2 is fixed and $\lambda \in \mathbb{R}$.

Similarly, if we take h as the density of the $\mathsf{Gamma}(a,1)$-distribution, and let $\theta = 1-\lambda$ and $\zeta(\theta) = -a\ln(1-\theta) = -a\ln\lambda$, we obtain the class of $\mathsf{Gamma}(a,\lambda)$ distributions, with a fixed and $\lambda > 0$. Note that in this case $\Theta = (-\infty, 1)$.

Starting from any pdf f_0, we can easily generate a natural exponential family of the form (A.12) in the following way: Let Θ be the largest interval for which the cumulant function ζ of f_0 exists. This includes $\theta = 0$, since f_0 is a pdf. Now define

$$f(x;\theta) = \mathrm{e}^{\theta x - \zeta(\theta)}\, f_0(x)\,. \tag{A.13}$$

Then $\{f(\cdot;\theta), \theta \in \Theta\}$ is a natural exponential family. We say that the family is obtained from f_0 by an *exponential twist* or *exponential change of measure* (ECM) with a *twisting* or *tilting* parameter θ.

Remark A.3.2 (Reparameterization) It may be useful to reparameterize a natural exponential family of the form (A.12) into the form (A.10). Let $X \sim f(\cdot;\theta)$. It is not difficult to see that

$$\mathbb{E}_\theta[X] = \zeta'(\theta) \quad \text{and} \quad \mathrm{Var}_\theta(X) = \zeta''(\theta)\,. \tag{A.14}$$

$\zeta'(\theta)$ is increasing in θ, since its derivative, $\zeta''(\theta) = \mathrm{Var}_\theta(X)$, is always greater than 0. Thus, we can reparameterize the family using the mean $v = \mathbb{E}_\theta[X]$. In particular, to the above natural exponential family there corresponds a family $\{g(\cdot;v)\}$ such that for each pair (θ, v) satisfying $\zeta'(\theta) = v$ we have $g(x;v) = f(x;\theta)$.

■ **EXAMPLE A.3**

Consider the second case in Example A.2. Note that we constructed in fact a natural exponential family $\{f(\cdot;\theta), \theta \in (-\infty,1)\}$ by exponentially twisting the $\mathsf{Gamma}(\alpha,1)$ distribution, with density $f_0(x) = x^{\alpha-1}\mathrm{e}^{-x}/\Gamma(a)$. We have $\zeta'(\theta) = \alpha/(1-\theta) = v$. This leads to the reparameterized density

$$g(x;v) \;=\; \exp\left(\theta x + \alpha\ln(1-\theta)\right) f_0(x) \;=\; \frac{\exp\left(-\frac{\alpha}{v}\,x\right)\left(\frac{\alpha}{v}\right)^{\alpha} x^{\alpha-1}}{\Gamma(\alpha)}\,,$$

corresponding to the $\mathsf{Gamma}(\alpha, \alpha v^{-1})$ distribution, $v > 0$.

CE Updating Formulas for Exponential Families

We now obtain an *analytic* formula for a general one-parameter exponential family. Let $X \sim f(x;u)$ for some nominal reference parameter u. For simplicity, assume that $\mathbb{E}_u[H(X)] > 0$ and that X is nonconstant. Let $f(x;u)$ be a member of a one-parameter exponential family $\{f(x;v)\}$. Suppose the parameterization $\eta = \psi(v)$ puts the family in canonical form. That is,

$$f(x;v) = g(x;\eta) = \mathrm{e}^{\eta x - \zeta(\eta)} h(x)\,.$$

Moreover, let us assume that v corresponds to the expectation of X. This can always be established by reparameterization; see Remark A.3.2. Note that, in particular, $v = \zeta'(\eta)$. Let $\theta = \psi(u)$ correspond to the nominal reference parameter. Since $\max_v \mathbb{E}_u[H(X) \ln f(X;v)] = \max_\eta \mathbb{E}_\theta[H(X) \ln g(X;\eta)]$, we may obtain the optimal solution v^* to (5.61) by finding, as in (5.62), the solution η^* to

$$\mathbb{E}_\theta \left[H(X) \frac{d}{d\eta} \ln g(X;\eta) \right] = 0$$

and putting $v^* = \psi^{-1}(\eta^*)$. Since $(\ln g(X;\eta))' = x - \zeta'(\eta)$, and $\zeta'(\eta) = v$, we see that v^* is given by the solution of $\mathbb{E}_u[H(X)\,(-v + X)] = 0$. Hence v^* is given by

$$v^* = \frac{\mathbb{E}_u[H(X)\,X]}{\mathbb{E}_u[H(X)]} = \frac{\mathbb{E}_w[H(X)\,W(X;u,w)\,X]}{\mathbb{E}_w[H(X)\,W(X;u,w)]} \tag{A.15}$$

for any reference parameter w. It is not difficult to check that v^* is indeed a unique global maximum of $D(v) = \mathbb{E}_u[H(X) \ln f(X;v)]$. The corresponding estimator $\widehat{v}$ of v^* in (A.15) is

$$\widehat{v} = \frac{\sum_{i=1}^N H(X_i)\,W(X_i;u,w)\,X_i}{\sum_{i=1}^N H(X_i) W(X_i;u,w)}, \tag{A.16}$$

where $X_1, \ldots, X_N$ is a random sample from the density $f(\cdot;w)$.

A similar explicit formula can be found for the case where $\mathbf{X} = (X_1, \ldots, X_n)$ is a vector of *independent* random variables such that each component X_j belongs to a one-parameter exponential family parameterized by the mean; that is, the density of each X_j is given by

$$f_j(x;u_j) = \mathrm{e}^{x\theta(u_j) - \zeta(\theta(u_j))}\, h_j(x),$$

where $\mathbf{u} = (u_1, \ldots, u_n)$ is the nominal reference parameter. It is easy to see that problem (5.64) under the independence assumption becomes "separable," that is, it reduces to n subproblems of the form above. Thus, we find that the optimal reference parameter vector $\mathbf{v}^* = (v_1^*, \ldots, v_n^*)$ is given as

$$v_j^* = \frac{\mathbb{E}_\mathbf{u}[H(\mathbf{X})\,X_j]}{\mathbb{E}_\mathbf{u}[H(\mathbf{X})]} = \frac{\mathbb{E}_\mathbf{w}[H(\mathbf{X})\,W(\mathbf{X};\mathbf{u},\mathbf{w})\,X_j]}{\mathbb{E}_\mathbf{w}[(\mathbf{X})\,W(\mathbf{X};\mathbf{u},\mathbf{w})]}. \tag{A.17}$$

Moreover, we can estimate the j-th component of $\mathbf{v}^*$ as

$$\widehat{v}_j = \frac{\sum_{i=1}^N H(\mathbf{X}_i)\,W(\mathbf{X}_i;\mathbf{u},\mathbf{w})X_{ij}}{\sum_{i=1}^N H(\mathbf{X}_i)\,W(\mathbf{X}_i;\mathbf{u},\mathbf{w})}, \tag{A.18}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is a random sample from the density $f(\cdot;\mathbf{w})$ and X_{ij} is the j-th component of $\mathbf{X}_i$.

A.4 SENSITIVITY ANALYSIS

The crucial issue in choosing a good importance sampling density $f(\mathbf{x};\mathbf{v})$ to estimate $\nabla^k \ell(\mathbf{u})$ via (7.16) is to ensure that the corresponding estimators have low variance. We consider this issue for the cases $k = 0$ and $k = 1$. For $k = 0$ this means minimizing the variance of $\widehat{\ell}(\mathbf{u};\mathbf{v})$ with respect to $\mathbf{v}$, which is equivalent to solving the minimization program

$$\min_{\mathbf{v}} \mathcal{L}^0(\mathbf{v};\mathbf{u}) = \min_{\mathbf{v}} \mathbb{E}_{\mathbf{v}}[H^2(\mathbf{X})\, W^2(\mathbf{X};\mathbf{u},\mathbf{v})] \,. \tag{A.19}$$

For the case $k = 1$, note that $\nabla\widehat{\ell}(\mathbf{u};\mathbf{v})$ is a vector rather than a scalar. To obtain a good reference vector $\mathbf{v}$, we now minimize the *trace of the associated covariance matrix*, which is equivalent to minimizing

$$\min_{\mathbf{v}} \mathcal{L}^1(\mathbf{v};\mathbf{u}) = \min_{\mathbf{v}} \mathbb{E}_{\mathbf{v}}\left[H^2(\mathbf{X})\, W^2(\mathbf{X};\mathbf{u},\mathbf{v})\ \mathrm{tr}\left(\mathbb{S}(\mathbf{u};\mathbf{x})\mathbb{S}(\mathbf{u};\mathbf{x})^T\right)\right], \tag{A.20}$$

where tr denotes the trace. For exponential families both optimization programs are *convex*, as demonstrated in the next proposition. To conform with our earlier notation for exponential families in Section A.3, we use $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ instead of $\mathbf{u}$ and $\mathbf{v}$, respectively.

A.4.1 Convexity Results

Proposition A.4.1 *Let* $\mathbf{X}$ *be a random vector from an* m*-parameter exponential family of the form (A.9). Then* $\mathcal{L}^k(\boldsymbol{\eta};\boldsymbol{\theta})$, $k = 0, 1$, *defined in (A.19) and (A.20), are convex functions of* $\boldsymbol{\eta}$.

Proof: Consider first the case $k = 0$. One has (see (7.23))

$$\mathcal{L}^0(\boldsymbol{\eta};\boldsymbol{\theta}) = c(\boldsymbol{\theta})^2 \int \frac{H^2(\mathbf{x})}{c(\boldsymbol{\eta})}\, \mathrm{e}^{(2\boldsymbol{\theta}-\boldsymbol{\eta})\cdot\mathbf{t}(\mathbf{x})} h(\mathbf{x})\mathrm{d}\mathbf{x} \,, \tag{A.21}$$

where

$$c(\boldsymbol{\eta})^{-1} = \int \mathrm{e}^{\boldsymbol{\eta}\cdot\mathbf{t}(\mathbf{z})} h(\mathbf{z})\, \mathrm{d}\mathbf{z} \,.$$

Substituting the above into (A.21) yields

$$\mathcal{L}^0(\boldsymbol{\eta};\boldsymbol{\theta}) = c(\boldsymbol{\theta})^2 \int\int H^2(\mathbf{x})\, \mathrm{e}^{2\boldsymbol{\theta}\cdot\mathbf{t}(\mathbf{x})+\boldsymbol{\eta}\cdot(\mathbf{t}(\mathbf{z})-\mathbf{t}(\mathbf{x}))} h(\mathbf{x})\, h(\mathbf{z})\, \mathrm{d}\mathbf{x}\, \mathrm{d}\mathbf{z} \,. \tag{A.22}$$

Now, for any linear function, $a(\boldsymbol{\eta})$ of $\boldsymbol{\eta}$, the function $\mathrm{e}^{a(\boldsymbol{\eta})}$ is convex. Since $H^2(\mathbf{x})$ is nonnegative, it follows that for any fixed $\boldsymbol{\theta}$, $\mathbf{x}$, and $\mathbf{z}$, the function under the integral sign in (A.22) is convex in $\boldsymbol{\eta}$. This implies the convexity of $\mathcal{L}^0(\boldsymbol{\eta};\boldsymbol{\theta})$.

The case $k = 1$ follows in exactly the same way, noting that the trace $\mathrm{tr}\left(\mathbb{S}(\boldsymbol{\theta};\mathbf{x})\mathbb{S}(\boldsymbol{\theta};\mathbf{x})^T\right)$ is a nonnegative function for $\mathbf{x}$ for any $\boldsymbol{\theta}$. □

Remark A.4.1 Proposition A.4.1 can be extended to the case where

$$\ell(\mathbf{u}) = \varphi(\ell_1(\mathbf{u}), \ldots, \ell_k(\mathbf{u}))$$

and

$$\ell_i(\mathbf{u}) = \mathbb{E}_{\mathbf{u}}[H_i(\mathbf{X})] = \mathbb{E}_{\mathbf{v}}[H_i(\mathbf{X})W(\mathbf{X};\mathbf{u};\mathbf{v})] = \mathbb{E}_{\mathbf{v}}[H_i W], \quad i = 1, \ldots, k \,.$$

Here the $\{H_i(\mathbf{X})\}$ are sample functions associated with the same random vector $\mathbf{X}$ and $\varphi(\cdot)$ is a real-valued differentiable function. We prove its validity for the case $k = 2$. In this case, the estimators of $\ell(\mathbf{u})$ can be written as

$$\widehat{\ell}(\mathbf{u};\mathbf{v}) = \varphi(\widehat{\ell}_1(\mathbf{u};\mathbf{v}), \widehat{\ell}_2(\mathbf{u};\mathbf{v})) \,,$$

where $\widehat{\ell}_1(\mathbf{u};\mathbf{v})$ and $\widehat{\ell}_2(\mathbf{u};\mathbf{v})$ are the usual importance sampling estimators of $\ell_1(\mathbf{u})$ and $\ell_2(\mathbf{u})$, respectively. By virtue of the delta method (see Problem 7.11), $N^{1/2}(\widehat{\ell}(\mathbf{u};\mathbf{v})-\ell(\mathbf{u}))$ is asymptotically normal, with mean 0 and variance

$$\begin{aligned}\sigma^2(\mathbf{v};\mathbf{u}) &= a^2\,\mathrm{Var}_{\mathbf{v}}(H_1W)+b^2\,\mathrm{Var}_{\mathbf{v}}(H_2W)+2\,a\,b\,\mathrm{Cov}_{\mathbf{v}}\,(H_1W,H_2W)\\ &= \mathbb{E}_{\mathbf{v}}\left[(aH_1+bH_2)^2\,W^2\right]+R(\mathbf{u})\,. \qquad \text{(A.23)}\end{aligned}$$

Here $R(\mathbf{u})$ consists of the remaining terms that are independent of $\mathbf{v}$, $a=\partial\varphi(x_1,x_2)/\partial x_1$ and $b=\partial\varphi(x_1,x_2)/\partial x_2$ at $(x_1,x_2)=(\ell_1(\mathbf{u}),\ell_2(\mathbf{u}))$. For example, for $\varphi(x_1,x_2)=x_1/x_2$, one gets $a=1/\ell_2(\mathbf{u})$ and $b=-\ell_1(\mathbf{u})/\ell_2(\mathbf{u})^2$.

The convexity of $\sigma^2(\mathbf{v};\mathbf{u})$ in $\mathbf{v}$ now follows similarly to the proof of Proposition A.4.1.

A.4.2 Monotonicity Results

Consider optimizing the functions $\mathcal{L}^k(\mathbf{v};\mathbf{u})$, $k=0,1$ in (A.19) and (A.20) with respect to $\mathbf{v}$. Let $\mathbf{v}^*(k)$ be the optimal solutions for $k=0,1$. The following proposition states that the optimal reference parameter always leads to a "fatter" tail for $f(\mathbf{x};\mathbf{v}^*)$ than that of the original pdf $f(\mathbf{x};\mathbf{u})$. This important phenomenon is the driving force for all of our beautiful results in this book, as well as for preventing the degeneracy of our importance sampling estimates. For simplicity, the result is given for the gamma distribution only. Similar results can be established with respect to some other parameters of the exponential family and for the CE approach.

Proposition A.4.2 *Let $X\sim\mathsf{Gamma}(\alpha,u)$. Suppose that $H^2(x)$ is a monotonically increasing function on the interval $[0,\infty)$. Then*

$$v^*(k)<u,\ k=0,1\,. \qquad \text{(A.24)}$$

Proof: The proof will be given for $k=0$ only. The proof for $k=1$, using the trace operator, is similar. To simplify the notation, we write $\mathcal{L}(v)$ for $\mathcal{L}^0(v;u)$.

Since $\mathcal{L}(v)$ is convex, it suffices to prove that its derivative with respect to v is positive at $v=u$. To this end, represent $\mathcal{L}(v)$ as

$$\mathcal{L}(v)=c\int_0^\infty v^{-\alpha}H^2(x)\,x^{\alpha-1}\mathrm{e}^{-(2u-v)x}\,\mathrm{d}x\,,$$

where the constant $c=u^{2\alpha}\Gamma(\alpha)^{-1}$ is independent of v. Differentiating $\mathcal{L}(v)$ above with respect to v at $v=u$, one has

$$\mathcal{L}'(v)|_{v=u}=\mathcal{L}'(u)=c\int_0^\infty (x-\alpha u^{-1})\,u^{-\alpha}H^2(x)\,x^{\alpha-1}\,\mathrm{e}^{-ux}\,\mathrm{d}x\,.$$

Integrating by parts yields

$$\begin{aligned}\mathcal{L}'(u) &= \lim_{R\to\infty} -c\,u^{-\alpha-1}R^\alpha\mathrm{e}^{-uR}\,H^2(R)+c\,u^{-\alpha-1}\int_0^\infty x^\alpha\,\mathrm{e}^{-ux}\,\mathrm{d}H^2(x)\\ &= c\,u^{-\alpha-1}\int_0^\infty x^\alpha\,\mathrm{e}^{-ux}\,\mathrm{d}H^2(x)\,, \qquad \text{(A.25)}\end{aligned}$$

provided $H^2(R)R^\alpha\exp(-uR)$ tends to 0 as $R\to\infty$. Finally, since $H^2(x)$ is monotonically increasing in x, we conclude that the integral (A.25) is positive, and consequently, $\mathcal{L}'(u)>0$. This fact, and the convexity of $\mathcal{L}(v)$, imply that $v^*(0)<u$. □

Proposition A.4.2 can be extended to the multidimensional gamma distribution, as well as to some other exponential family distributions. For details see [5].

A.5 A SIMPLE CE ALGORITHM FOR OPTIMIZING THE PEAKS FUNCTION

The following Matlab code provides a simple implementation of a CE algorithm to solve the peaks function; see Example 8.12 on page 268.

```
n = 2;                                  % dimension
mu = [-3,-3]; sigma = 3*ones(1,n); N = 100; eps = 1E-5; rho = 0.1;

while max(sigma) > eps
   X = randn(N,n)*diag(sigma)+ mu(ones(N,1),:);
   SX= S(X);                            %Compute the performance
   sortSX = sortrows([X, SX],n+1);
   Elite = sortSX((1-rho)*N:N,1:n); % elite samples
   mu = mean(Elite,1);                  % take sample mean row-wise
   sigma = std(Elite,1);                % take sample st.dev. row-wise
   [S(mu),mu,max(sigma)]                % output the result
end

function out = S(X)
out =  3*(1-X(:,1)).^2.*exp(-(X(:,1).^2) - (X(:,2)+1).^2) ...
   - 10*(X(:,1)/5 - X(:,1).^3 - X(:,2).^5).*exp(-X(:,1).^2-X(:,2).^2) ...
   - 1/3*exp(-(X(:,1)+1).^2 - X(:,2).^2);
end
```

A.6 DISCRETE-TIME KALMAN FILTER

Consider the hidden Markov model

$$\begin{aligned} X_t &= A\,X_{t-1} + \varepsilon_{1t} \\ Y_t &= B\,X_t + \varepsilon_{2t}, \qquad t = 1, 2, \ldots, \end{aligned} \tag{A.26}$$

where A and B are matrices (B does not have to be a square matrix). We adopt the notation of Section 5.7.1. The initial state X_0 is assumed to be $\mathsf{N}(\mu_0, \Sigma_0)$ distributed. The objective is to obtain the filtering pdf $f(x_t \mid \mathbf{y}_{1:t})$ and the *predictive* pdf $f(x_t \mid \mathbf{y}_{1:t-1})$. Observe that the joint pdf of $\mathbf{X}_{1:t}$ and $\mathbf{Y}_{1:t}$ must be Gaussian, since these random vectors are linear transformations of independent standard Gaussian random variables. It follows that $f(x_t \mid \mathbf{y}_{1:t}) \sim \mathsf{N}(\mu_t, \Sigma_t)$ for some mean vector μ_t and covariance matrix Σ_t. Similarly, $f(x_t \mid \mathbf{y}_{1:t-1}) \sim \mathsf{N}(\widetilde{\mu}_t, \widetilde{\Sigma}_t)$ for some mean vector $\widetilde{\mu}_t$ and covariance matrix $\widetilde{\Sigma}_t$. We wish to compute μ_t, $\widetilde{\mu}_t$, Σ_t and $\widetilde{\Sigma}_t$ recursively. The argument goes as follows: by assumption, $(X_{t-1} \mid \mathbf{y}_{1:t-1}) \sim \mathsf{N}(\mu_{t-1}, \Sigma_{t-1})$. Combining this with the fact that $X_t = A\,X_{t-1} + \varepsilon_{1t}$ yields

$$(X_t \mid \mathbf{y}_{1:t-1}) \sim \mathsf{N}(A\,\mu_{t-1},\; A\Sigma_{t-1}A^T + C_1)\,.$$

In other words,

$$\begin{aligned} \widetilde{\mu}_t &= A\,\mu_{t-1}, \\ \widetilde{\Sigma}_t &= A\Sigma_{t-1}A^T + C_1\,. \end{aligned} \tag{A.27}$$

Next, we determine the joint pdf of X_t and Y_t given $\mathbf{Y}_{1:t-1} = \mathbf{y}_{1:t-1}$. Decomposing $\widetilde{\Sigma}_t$ and C_2 as $\widetilde{\Sigma}_t = RR^T$ and $C_2 = QQ^T$, respectively (e.g., via the Cholesky square root

method), we can write (see (1.23))

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix} \Bigg| \mathbf{y}_{1:t-1} = \begin{pmatrix} \widetilde{\mu}_t \\ B\widetilde{\mu}_t \end{pmatrix} + \begin{pmatrix} R & 0 \\ BR & Q \end{pmatrix} \begin{pmatrix} U \\ V \end{pmatrix},$$

where, conditional on $Y_{t-1} = \mathbf{y}_{1:t-1}$, U and V are independent standard normal random vectors. The corresponding covariance matrix is

$$\begin{pmatrix} R & 0 \\ BR & Q \end{pmatrix} \begin{pmatrix} R^T & R^T B^T \\ 0 & Q^T \end{pmatrix} = \begin{pmatrix} RR^T & RR^T B^T \\ BRR^T & BRR^T B^T + QQ^T \end{pmatrix},$$

so that we have

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix} \Bigg| \mathbf{y}_{1:t-1} \sim \mathsf{N}\left(\begin{pmatrix} \widetilde{\mu}_t \\ B\widetilde{\mu}_t \end{pmatrix}, \begin{pmatrix} \widetilde{\Sigma}_t & \widetilde{\Sigma}_t B^T \\ B\widetilde{\Sigma}_t & B\widetilde{\Sigma}_t B^T + C_2 \end{pmatrix} \right) \tag{A.28}$$

(note that $\widetilde{\Sigma}_t$ is symmetric).

The result (A.28) enables us to find the conditional pdf $f(x_t \mid \mathbf{y}_t)$ with the aid of the following general result (see Problem A.2 below for a proof): If

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathsf{N}\left(\begin{pmatrix} m_1 \\ m_2 \end{pmatrix}, \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \right),$$

then

$$(X \mid Y = y) \sim \mathsf{N}\left(m_1 + S_{12} S_{22}^{-1}(y - m_2),\ S_{11} - S_{12} S_{22}^{-1} S_{12}^T\right). \tag{A.29}$$

Because $f(x_t \mid \mathbf{y}_{1:t}) = f(x_t \mid \mathbf{y}_{1:t-1}, y_t)$, an immediate consequence of (A.28) and (A.29) is

$$\begin{aligned} \mu_t &= \widetilde{\mu}_t + \widetilde{\Sigma}_t B^T (B\widetilde{\Sigma}_t B^T + C_2)^{-1}(y_t - B\widetilde{\mu}_t), \\ \Sigma_t &= \widetilde{\Sigma}_t - \widetilde{\Sigma}_t B^T (B\widetilde{\Sigma}_t B^T + C_2)^{-1} B\widetilde{\Sigma}_t. \end{aligned} \tag{A.30}$$

Updating formulas (A.27) and (A.30) form the (discrete-time) *Kalman filter*. Starting with some known μ_0 and Σ_0, one determines $\widetilde{\mu}_1$ and $\widetilde{\Sigma}_1$, then μ_1 and Σ_1, and so on. Notice that $\widetilde{\Sigma}_t$ and Σ_t do not depend on the observations $y_1, y_2, \ldots$ and can therefore be determined *off-line*. The Kalman filter discussed above can be extended in many ways, for example by including control variables and time-varying parameter matrices. The nonlinear filtering case is often dealt with by linearizing the state and observation equations via a Taylor expansion. This leads to an approximative method called the *extended Kalman filter*.

A.7 BERNOULLI DISRUPTION PROBLEM

As an example of a finite-state hidden Markov model, we consider the following *Bernoulli disruption problem*. In Example 6.8 a similar type of "changepoint" problem is discussed in relation to the Gibbs sampler. However, the crucial difference is that in the present case the detection of the changepoint can be done *sequentially*.

Let $Y_1, Y_2, \ldots$ be Bernoulli random variables and let T be a geometrically distributed random variable with parameter r. Conditional upon T the $\{Y_i\}$ are mutually independent, and $Y_1, Y_2, \ldots, Y_{T-1}$ all have a success probability a, whereas $Y_T, Y_{T+1}, \ldots$ all have a success probability b. Thus, T is the change or disruption point. Suppose that T cannot

be observed, but only $\{Y_t\}$. We wish to decide if the disruption has occurred based on the outcome $\mathbf{y}_{1:t} = (y_1, \ldots, y_t)$ of $\mathbf{Y}_{1:t} = (Y_1, \ldots, Y_t)$. An example of the observations is depicted in Figure A.1, where the dark lines indicate the times of successes ($Y_i = 1$).

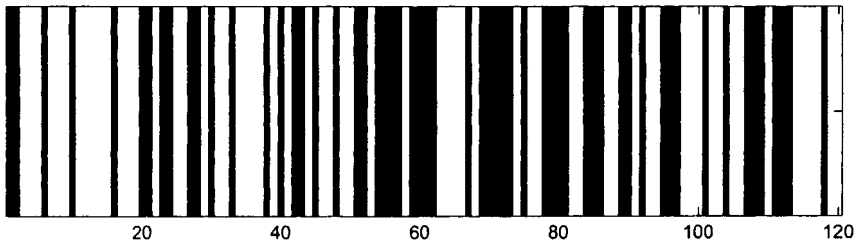


Figure A.1 The observations for the disruption problem.

The situation can be described via the HMM illustrated in Figure A.2. Namely, let $\{X_t, t = 0, 1, 2, \ldots\}$ be a Markov chain with state space $\{0, 1\}$, transition matrix

$$P = \begin{pmatrix} 1-r & r \\ 0 & 1 \end{pmatrix},$$

and initial state $X_0 = 0$. Then the objective is to find $\mathbb{P}(T \leqslant t \,|\, \mathbf{Y}_{1:t} = \mathbf{y}_{1:t}) = \mathbb{P}(X_t = 1 \,|\, \mathbf{Y}_t = \mathbf{y}_{1:t})$.

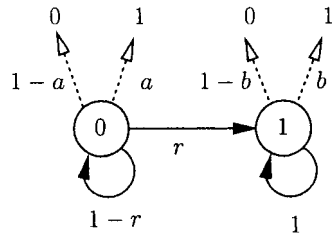


Figure A.2 The HMM diagram for the disruption problem.

This can be done efficiently by introducing

$$\alpha_t(j) = \mathbb{P}(X_t = j,\ \mathbf{Y}_{1:t} = \mathbf{y}_{1:t})\,.$$

By conditioning on X_{t-1} we have

$$\begin{aligned}
\alpha_t(j) &= \sum_i \mathbb{P}(X_t = j, X_{t-1} = i, \mathbf{Y}_{1:t} = \mathbf{y}_{1:t}) \\
&= \sum_i \mathbb{P}(X_t = j, Y_t = y_t \,|\, X_{t-1} = i, \mathbf{Y}_{1:t-1} = \mathbf{y}_{1:t-1})\, \alpha_{t-1}(i) \\
&= \sum_i \mathbb{P}(X_t = j, Y_t = y_t \,|\, X_{t-1} = i)\, \alpha_{t-1}(i). \\
&= \sum_i \mathbb{P}(Y_t = y_t \,|\, X_t = j)\, \mathbb{P}(X_t = j \,|\, X_{t-1} = i)\, \alpha_{t-1}(i)\,.
\end{aligned}$$

In particular, we find the recurrence relation

$$\alpha_t(0) = a_{0\,y_t}\,(1-r)\,\alpha_{t-1}(0) \quad \text{and} \quad \alpha_t(1) = a_{1\,y_t}\{r\,\alpha_{t-1}(0) + \alpha_{t-1}(1)\}\,,$$

with $a_{ij} = \mathbb{P}(Y = j \,|\, X = i)$, $i, j \in \{0, 1\}$ (thus, $a_{00} = 1 - a$, $a_{01} = a$, $a_{10} = 1 - b$, $a_{11} = b$), and initial values

$$\alpha_1(0) = a^{y_1}(1-a)^{1-y_1}(1-r) \quad \text{and} \quad \alpha_1(1) = b^{y_1}(1-b)^{1-y_1} r\,.$$

In Figure A.3 a plot is given of the probability $\mathbb{P}(X_t = 1 \,|\, \mathbf{Y}_{1:t} = \mathbf{y}_{1:t}) = \alpha_t(1)/(\alpha_t(1) + \alpha_t(2))$, as a function of t, for a test case with $a = 0.4$, $b = 0.6$, and $r = 0.01$. In this particular case $T = 49$. We see a dramatic change in the graph after the disruption takes effect.

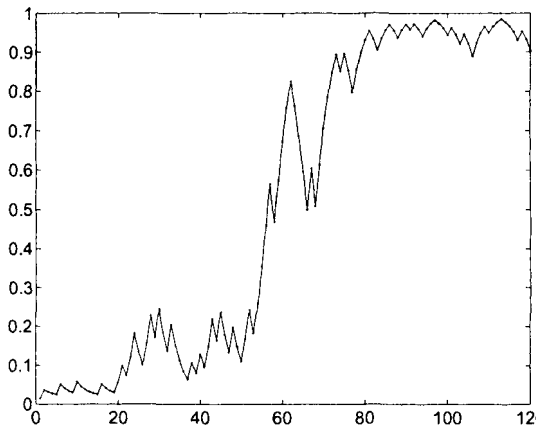


Figure A.3 The probability $\mathbb{P}(X_t = 1 \,|\, \mathbf{Y}_{1:t} = \mathbf{y}_{1:t})$ as a function of t.

A.8 COMPLEXITY OF STOCHASTIC PROGRAMMING PROBLEMS

Consider the following optimization problem:

$$\ell^* = \min_{\mathbf{u} \in \mathscr{U}} \ell(\mathbf{u}) = \min_{\mathbf{u} \in \mathscr{U}} \mathbb{E}_f[H(\mathbf{X}; \mathbf{u})]\,, \tag{A.31}$$

where it is assumed that $\mathbf{X}$ is a random vector with known pdf f having support $\mathscr{X} \subset \mathbb{R}^n$, and $H(\mathbf{X}; \mathbf{u})$ is the sample function depending on $\mathbf{X}$ and the decision vector $\mathbf{u} \in \mathbb{R}^m$.

As an example, consider a two-stage stochastic programming problem with recourse, which is an optimization problem that is divided into two stages. At the first stage, one has to make a decision on the basis of some available information. At the second stage, after a realization of the uncertain data becomes known, an optimal second-stage decision is made. Such a stochastic programming problem can be written in the form (A.31), with $H(\mathbf{X}; \mathbf{u})$ being the optimal value of the second-stage problem.

We now discuss the issue of how difficult it is to solve a stochastic program of type (A.31). We should expect that this problem is at least as difficult as minimizing $\ell(\mathbf{u})$, $\mathbf{u} \in \mathscr{U}$ in the case where $\ell(\mathbf{u})$ is given *explicitly*, say by a closed-form analytic expression or, more generally, by an "oracle" capable of computing the values and the derivatives of

$\ell(\mathbf{u})$ at every given point. As far as problems of minimization of $\ell(\mathbf{u})$, $\mathbf{u} \in \mathscr{U}$, with an explicitly given objective are concerned, the solvable case is known: this is the convex programming case. That is, $\mathscr{U}$ is a closed convex set and $\ell : \mathscr{U} \to \mathbb{R}$ is a convex function. It is known that generic convex programming problems satisfying mild computability and boundedness assumptions can be solved in polynomial time. In contrast to this, typical nonconvex problems turn out to be NP-hard.

We should also stress that a claim that "such and such problem is difficult" relates to a *generic* problem and does *not* imply that the problem has no solvable particular cases. When speaking about conditions under which the stochastic program (A.31) is efficiently solvable, it makes sense to assume that $\mathscr{U}$ is a closed convex set and $\ell(\cdot)$ is convex on $\mathscr{U}$. We gain from a technical viewpoint (and do not lose much from a practical viewpoint) by assuming $\mathscr{U}$ to be bounded. These assumptions, plus mild technical conditions, would be sufficient to make (A.31) easy (manageable) if $\ell(\mathbf{u})$ were given explicitly. However, in stochastic programming, it makes no sense to assume that we can compute efficiently the expectation in (A.31), thus arriving at an explicit representation of $\ell(\mathbf{u})$. If this were the case, there would be no necessity to treat (A.31) as a stochastic program.

We argue now that stochastic programming problems of the form (A.31) can be solved reasonably efficiently by using Monte Carlo sampling techniques, provided that the probability distribution of the random data is not "too bad" and certain general conditions are met. In this respect, we should explain what we mean by "solving" stochastic programming problems. Let us consider, for example, two-stage linear stochastic programming problems with recourse. Such problems can be written in the form (A.31) with

$$\mathscr{U} = \{\mathbf{u} : A\mathbf{u} = \mathbf{b},\ \mathbf{u} \geqslant \mathbf{0}\} \text{ and } H(\mathbf{X}; \mathbf{u}) = \langle \mathbf{c}, \mathbf{u} \rangle + Q(\mathbf{X}; \mathbf{u}) ,$$

where $\langle \mathbf{c}, \mathbf{u} \rangle$ is the cost of the first-stage decision and $Q(\mathbf{X}; \mathbf{u})$ is the optimal value of the second-stage problem:

$$\min_{\mathbf{y} \geqslant \mathbf{0}} \langle \mathbf{q}, \mathbf{y} \rangle \text{ subject to } \mathbf{T}\mathbf{u} + \mathbf{W}\mathbf{y} \geqslant \mathbf{h} . \tag{A.32}$$

Here, $\langle \cdot, \cdot \rangle$ denotes the inner product. $\mathbf{X}$ is a vector whose elements are composed from elements of vectors $\mathbf{q}$ and $\mathbf{h}$ and matrices $\mathbf{T}$ and $\mathbf{W}$, which are assumed to be random.

If we assume that the random data vector $\mathbf{X} = (\mathbf{q}, \mathbf{W}, \mathbf{T}, \mathbf{h})$ takes K different values (called *scenarios*) $\{\mathbf{X}_k, k = 1, \ldots, K\}$, with respective probabilities $\{p_k, k = 1, \ldots, K\}$, then the obtained two-stage problem can be written as one large linear programming problem:

$$\begin{array}{ll} \min\limits_{\mathbf{u}, \mathbf{y}_1, \ldots, \mathbf{y}_K} & \langle \mathbf{c}, \mathbf{u} \rangle + \sum_{k=1}^{K} p_k \langle \mathbf{q}_k, \mathbf{y}_k \rangle \\ \text{s.t} & \mathbf{A}\mathbf{u} = \mathbf{b},\ \mathbf{T}_k\mathbf{u} + \mathbf{W}_k\mathbf{y}_k \geqslant \mathbf{h}_k,\ k = 1, \ldots, K, \\ & \mathbf{u} \geqslant \mathbf{0},\ \mathbf{y}_k \geqslant \mathbf{0},\ k = 1, \ldots, K. \end{array} \tag{A.33}$$

If the number of scenarios K is not too large, then the above linear programming problem (A.33) can be solved accurately in a reasonable period of time. However, even a crude discretization of the probability distribution of $\mathbf{X}$ typically results in an exponential growth of the number of scenarios with the increase of the dimension of $\mathbf{X}$. Suppose, for example, that the components of the random vector $\mathbf{X}$ are mutually independently distributed, each having a small number r of possible realizations. Then the size of the corresponding input data grows linearly in n (and r), while the number of scenarios $K = r^n$ grows exponentially.

We would like to stress that from a practical point of view, it does not make sense to try to solve a stochastic programming problem with high precision. A numerical error resulting from an inaccurate estimation of the involved probability distributions, modeling errors,

and so on, can be far bigger than the optimization error. We argue now that two-stage stochastic problems can be solved efficiently with reasonable accuracy, provided that the following conditions are met:

(a) The feasible set $\mathscr{U}$ is fixed (deterministic).

(b) For all $\mathbf{u} \in \mathscr{U}$ and $\mathbf{X} \in \mathscr{X}$, the objective function $H(\mathbf{X}; \mathbf{u})$ is real-valued.

(c) The considered stochastic programming problem can be solved efficiently (by a deterministic algorithm) if the number of scenarios is not too large.

When applied to two-stage stochastic programming, the above conditions (a) and (b) mean that the recourse is relatively complete and the second-stage problem is bounded from below. Note that it is said that the recourse is *relatively complete*, if for every $\mathbf{u} \in \mathscr{U}$ and every possible realization of random data, the second-stage problem is feasible. The above condition (c) certainly holds in the case of two-stage *linear* stochastic programming with recourse.

In order to proceed, let us consider the following Monte Carlo sampling approach. Suppose that we can generate an iid random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from $f(\mathbf{x})$, and we can estimate the expected value function $\ell(\mathbf{u})$ by the sample average

$$\widehat{\ell}(\mathbf{u}) = \frac{1}{N} \sum_{j=1}^{N} H(\mathbf{X}_j; \mathbf{u}) . \tag{A.34}$$

Note that $\widehat{\ell}$ depends on the sample size N and on the generated sample, and in that sense is random. Consequently, we approximate the true problem (A.31) by the following approximated one:

$$\min_{\mathbf{u} \in \mathscr{U}} \widehat{\ell}(\mathbf{u}) . \tag{A.35}$$

We refer to (A.35) as the *stochastic counterpart* or *sample average approximation* problem. The optimal value $\widehat{\ell}^*$ and the set $\widehat{\mathscr{U}}^*$ of optimal solutions of the stochastic counterpart problem (A.35) provide estimates of their true counterparts, ℓ^* and $\mathscr{U}^*$, of problem (A.31). It should be noted that once the sample is generated, $\widehat{\ell}(\mathbf{u})$ becomes a deterministic function and problem (A.35) becomes a stochastic programming problem with N scenarios $\mathbf{X}_1, \ldots, \mathbf{X}_N$ taken with equal probabilities $1/N$. It also should be mentioned that the stochastic counterpart method is *not* an algorithm. One still has to solve the obtained problem (A.35) by employing an appropriate (deterministic) algorithm.

By the law of large numbers (see Theorem 1.10.1) $\widehat{\ell}(\mathbf{u})$ converges (point-wise in $\mathscr{U}$) with probability 1 to $\ell(\mathbf{u})$ as N tends to infinity. Therefore, it is reasonable to expect for $\widehat{\ell}^*$ and $\widehat{\mathscr{U}}^*$ to converge to their counterparts of the true problem (A.31) with probability 1 as N tends to infinity. And indeed, such convergence can be proved under mild regularity conditions. However, for a fixed $\mathbf{u} \in \mathscr{U}$, convergence of $\widehat{\ell}(\mathbf{u})$ to $\ell(\mathbf{u})$ is notoriously slow. By the central limit theorem (see Theorem 1.10.2) it is of order $\mathcal{O}(N^{-1/2})$. The rate of convergence can be improved, sometimes significantly, by variance reduction methods. However, using Monte Carlo techniques, one cannot evaluate the expected value $\ell(\mathbf{u})$ very accurately.

The following analysis is based on the exponential bounds of the *large deviations* theory. Denote by $\mathscr{U}^\varepsilon$ and $\widehat{\mathscr{U}}^\varepsilon$ the sets of ε-optimal solutions of the true and stochastic counterpart problems, respectively, that is, $\bar{\mathbf{u}} \in \mathscr{U}^\varepsilon$ iff $\bar{\mathbf{u}} \in \mathscr{U}$ and $\ell(\bar{\mathbf{u}}) \leqslant \inf_{\mathbf{u} \in \mathscr{U}} \ell(\mathbf{u}) + \varepsilon$. Note that for $\varepsilon = 0$ the set $\mathscr{U}^0$ coincides with the set of the optimal solutions of the true problem.

Choose accuracy constants $\varepsilon > 0$ and $0 \leqslant \delta < \varepsilon$ and the confidence (significance) level $\alpha \in (0, 1)$. Suppose for the moment that the set $\mathscr{U}$ is finite, although its cardinality $|\mathscr{U}|$ can be very large. Then, using Cramér's large deviations theorem, it can be shown [4] that there exists a constant $\eta(\varepsilon, \delta)$ such that

$$N \geqslant \frac{1}{\eta(\varepsilon, \delta)} \ln\left(\frac{|\mathscr{U}|}{\alpha}\right) \tag{A.36}$$

guarantees that the probability of the event $\{\widehat{\mathscr{U}}^{\delta} \subset \mathscr{U}^{\varepsilon}\}$ is at least $1 - \alpha$. That is, for any N bigger than the right-hand side of (A.36), we are guaranteed that any δ-optimal solution of the corresponding stochastic counterpart problem provides an ε-optimal solution of the true problem with probability at least $1 - \alpha$. In other words, solving the stochastic counterpart problem with accuracy δ guarantees solving the true problem with accuracy ε with probability at least $1 - \alpha$.

The number $\eta(\varepsilon, \delta)$ in the estimate (A.36) is defined as follows. Consider a mapping $\pi : \mathscr{U} \setminus \mathscr{U}^{\varepsilon} \to \mathscr{U}$ such that $\ell(\pi(\mathbf{u})) \leqslant \ell(\mathbf{u}) - \varepsilon$ for all $\mathbf{u} \in \mathscr{U} \setminus \mathscr{U}^{\varepsilon}$. Such mappings do exist, although not uniquely. For example, any mapping $\pi : \mathscr{U} \setminus \mathscr{U}^{\varepsilon} \to \mathscr{U}^{0}$ satisfies this condition. The choice of such a mapping gives a certain flexibility to the corresponding estimate of the sample size. For $\mathbf{u} \in \mathscr{U}$, consider the random variable

$$Y_{\mathbf{u}} = H(\mathbf{X}; \pi(\mathbf{u})) - H(\mathbf{X}; \mathbf{u}) ,$$

its moment generating function $M_{\mathbf{u}}(t) = \mathbb{E}\left[e^{tY_{\mathbf{u}}}\right]$, and the large deviations *rate function*

$$I_{\mathbf{u}}(z) = \sup_{t \in \mathbb{R}} \left\{tz - \ln M_{\mathbf{u}}(t)\right\} .$$

Note that $I_{\mathbf{u}}(\cdot)$ is the conjugate of the function $\ln M_{\mathbf{u}}(\cdot)$ in the sense of convex analysis. Note also that, by construction of mapping $\pi(\mathbf{u})$, the inequality

$$\mu_{\mathbf{u}} = \mathbb{E}[Y_{\mathbf{u}}] = \ell(\pi(\mathbf{u})) - \ell(\mathbf{u}) \leqslant -\varepsilon \tag{A.37}$$

holds for all $\mathbf{u} \in \mathscr{U} \setminus \mathscr{U}^{\varepsilon}$. Finally, we define

$$\eta(\varepsilon, \delta) = \min_{\mathbf{u} \in \mathscr{U} \setminus \mathscr{U}^{\varepsilon}} I_{\mathbf{u}}(-\delta) . \tag{A.38}$$

Because of (A.37) and since $\delta < \varepsilon$, the number $I_{\mathbf{u}}(-\delta)$ is positive, provided that the probability distribution of $Y_{\mathbf{u}}$ is not too bad. Specifically, if we assume that the moment generating function $M_{\mathbf{u}}(t)$, of $Y_{\mathbf{u}}$, is finite-valued for all t in a neighborhood of 0, then the random variable $Y_{\mathbf{u}}$ has finite moments and $I_{\mathbf{u}}(\mu_{\mathbf{u}}) = I'(\mu_{\mathbf{u}}) = 0$, and $I''(\mu_{\mathbf{u}}) = 1/\sigma_{\mathbf{u}}^2$ where $\sigma_{\mathbf{u}}^2 = \mathrm{Var}[Y_{\mathbf{u}}]$. Consequently, $I_{\mathbf{u}}(-\delta)$ can be approximated by using the second-order Taylor expansion, as follows:

$$I_{\mathbf{u}}(-\delta) \approx \frac{(-\delta - \mu_{\mathbf{u}})^2}{2\sigma_{\mathbf{u}}^2} \geqslant \frac{(\varepsilon - \delta)^2}{2\sigma_{\mathbf{u}}^2} .$$

This suggests that one can expect the constant $\eta(\varepsilon, \delta)$ to be of order $(\varepsilon - \delta)^2$. And indeed, this can be ensured by various conditions. Consider the following ones.

(A1) *There exists a constant $\sigma > 0$ such that for any $\mathbf{u} \in \mathscr{U} \setminus \mathscr{U}^{\varepsilon}$, the moment generating function $M_{\mathbf{u}}^{*}(t)$ of the random variable $Y_{\mathbf{u}} - \mathbb{E}[Y_{\mathbf{u}}]$ satisfies*

$$M_{\mathbf{u}}^{*}(t) \leqslant \exp\left(\sigma^2 t^2/2\right), \quad \forall t \in \mathbb{R} . \tag{A.39}$$

Note that the random variable $Y_{\mathbf{u}} - \mathbb{E}[Y_{\mathbf{u}}]$ has zero mean. Moreover, if it has a normal distribution, with variance $\sigma_{\mathbf{u}}^2$, then its moment generating function is equal to the right-hand side of (A.39). Condition (A.39) means that the tail probabilities $\mathbb{P}\big(|H(\mathbf{X};\pi(\mathbf{u})) - H(\mathbf{X};\mathbf{u})| > t\big)$ are bounded from above by $\mathcal{O}(1)\exp\big(-t^2/(2\sigma_{\mathbf{u}}^2)\big)$. Note that by $\mathcal{O}(1)$ we denote generic absolute constants. This condition certainly holds if the distribution of the considered random variable has a bounded support. Condition (A.39) implies that $M_{\mathbf{u}}(t) \leqslant \exp(\mu_{\mathbf{u}} t + \sigma^2 t^2/2)$. It follows that

$$I_{\mathbf{u}}(z) \geqslant \sup_{t\in\mathbb{R}} \big\{tz - \mu_{\mathbf{u}} t - \sigma^2 t^2/2\big\} = \frac{(z-\mu_{\mathbf{u}})^2}{2\sigma^2}\,, \tag{A.40}$$

and hence, for any $\varepsilon > 0$ and $\delta \in [0, \varepsilon)$,

$$\eta(\varepsilon,\delta) \geqslant \frac{(-\delta-\mu_{\mathbf{u}})^2}{2\sigma^2} \geqslant \frac{(\varepsilon-\delta)^2}{2\sigma^2}\,. \tag{A.41}$$

It follows that, under assumption (A1), the estimate (A.36) can be written as

$$N \geqslant \frac{2\sigma^2}{(\varepsilon-\delta)^2} \ln\left(\frac{|\mathscr{U}|}{\alpha}\right)\,. \tag{A.42}$$

Remark A.8.1 Condition (A.39) can be replaced by a more general one,

$$M_{\mathbf{u}}^*(t) \leqslant \exp(\psi(t)), \quad \forall t \in \mathbb{R}\,, \tag{A.43}$$

where $\psi(t)$ is a convex even function with $\psi(0) = 0$. Then $\ln M_{\mathbf{u}}(t) \leqslant \mu_{\mathbf{u}} t + \psi(t)$ and hence $I_{\mathbf{u}}(z) \geqslant \psi^*(z - \mu_{\mathbf{u}})$, where ψ^* is the conjugate of the function ψ. It follows then that

$$\eta(\varepsilon,\delta) \geqslant \psi^*(-\delta-\mu_{\mathbf{u}}) \geqslant \psi^*(\varepsilon-\delta)\,. \tag{A.44}$$

For example, instead of assuming that the bound (A.39) holds for all $t \in \mathbb{R}$, we can assume that it holds for all t in a finite interval $[-a, a]$, where $a > 0$ is a given constant. That is, we can take $\psi(t) = \sigma^2 t/2$ if $|t| \leqslant a$ and $\psi(t) = +\infty$ otherwise. In that case, $\psi^*(z) = z^2/(2\sigma^2)$ for $|z| \leqslant a\sigma^2$ and $\psi^*(z) = a|z| - a^2\sigma^2/2$ for $|z| > a\sigma^2$.

A key feature of the estimate (A.42) is that the required sample size N depends *logarithmically* both on the size of the feasible set $\mathscr{U}$ and on the significance level α. The constant σ, postulated in assumption (A1), measures, in some sense, the variability of the considered problem. For, say, $\delta = \varepsilon/2$, the right-hand side of the estimate (A.42) is proportional to $(\sigma/\varepsilon)^2$. For Monte Carlo methods, such dependence on σ and ε seems to be unavoidable. In order to see this, consider a simple case when the feasible set $\mathscr{U}$ consists of just two elements: $\mathscr{U} = \{u_1, u_2\}$, with $\ell(u_2) - \ell(u_1) > \varepsilon > 0$. By solving the corresponding stochastic counterpart problem, we can ensure that u_1 is the ε-optimal solution if $\widehat{\ell}(u_2) - \widehat{\ell}(u_1) > 0$. If the random variable $H(X;u_2) - H(X;u_1)$ has a normal distribution with mean $\mu = \ell(u_2) - \ell(u_1)$ and variance σ^2, then $\widehat{\ell}(u_2) - \widehat{\ell}(u_1) \sim \mathsf{N}(\mu, \sigma^2/N)$ and the probability of the event $\{\widehat{\ell}(u_2) - \widehat{\ell}(u_1) > 0\}$ (that is, of the correct decision) is $\Phi(\mu\sqrt{N}/\sigma)$, where Φ is the cdf of $\mathsf{N}(0,1)$. We have that $\Phi(\varepsilon\sqrt{N}/\sigma) < \Phi(\mu\sqrt{N}/\sigma)$, and in order to make the probability of the incorrect decision less than α, we have to take the sample size $N > z_{1-\alpha}^2\,\sigma^2/\varepsilon^2$, where $z_{1-\alpha}$ is the $(1-\alpha)$-quantile of the standard normal distribution. Even if $H(X;u_2) - H(X;u_1)$ is not normally distributed, the sample size of order σ^2/ε^2 could be justified asymptotically, say by applying the central limit theorem.

Let us also consider the following simplified variant of the estimate (A.42). Suppose that:

(A2) *There is a positive constant C such that the random variable $\mathbf{Y}_{\mathbf{u}}$ is bounded in absolute value by a constant C for all $\mathbf{u} \in \mathscr{U} \setminus \mathscr{U}^{\varepsilon}$.*

Under assumption (A2) we have that for any $\varepsilon > 0$ and $\delta \in [0, \varepsilon]$:

$$I_{\mathbf{u}}(-\delta) \geqslant \mathcal{O}(1)\frac{(\varepsilon - \delta)^2}{C^2}, \quad \text{for all } \mathbf{u} \in \mathscr{U} \setminus \mathscr{U}^{\varepsilon}, \tag{A.45}$$

and hence $\eta(\varepsilon, \delta) \geqslant \mathcal{O}(1)(\varepsilon - \delta)^2/C^2$. Consequently, the bound (A.36) for the sample size that is required to solve the true problem with accuracy $\varepsilon > 0$ and probability at least $1 - \alpha$, by solving the stochastic counterpart problem with accuracy $\delta = \varepsilon/2$, takes the form

$$N \geqslant \mathcal{O}(1)\left(\frac{C}{\varepsilon}\right)^2 \ln\left(\frac{|\mathscr{U}|}{\alpha}\right). \tag{A.46}$$

Now let $\mathscr{U}$ be a bounded, not necessarily a finite, subset of $\mathbb{R}^m$ of diameter

$$D = \sup_{\mathbf{u}', \mathbf{u} \in \mathscr{U}} \|\mathbf{u}' - \mathbf{u}\|.$$

Then for $\tau > 0$, we can construct a set $\mathscr{U}_\tau \subset \mathscr{U}$ such that for any $\mathbf{u} \in \mathscr{U}$ there is $\mathbf{u}' \in \mathscr{U}_\tau$ satisfying $\|\mathbf{u} - \mathbf{u}'\| \leqslant \tau$, and $|\mathscr{U}_\tau| = (\mathcal{O}(1)D/\tau)^m$.

Suppose next that the following condition holds:

(A3) *There exists a constant $\sigma > 0$ such that for any $\mathbf{u}', \mathbf{u} \in \mathscr{U}$ the moment generating function $M_{\mathbf{u}',\mathbf{u}}(t)$, of random variable $H(\mathbf{X}; \mathbf{u}') - H(\mathbf{X}; \mathbf{u}) - \mathbb{E}[H(\mathbf{X}; \mathbf{u}') - H(\mathbf{X}; \mathbf{u})]$, satisfies*

$$M_{\mathbf{u}',\mathbf{u}}(t) \leqslant \exp\left(\sigma^2 t^2/2\right), \quad \forall t \in \mathbb{R}. \tag{A.47}$$

The above assumption (A3) is slightly stronger than assumption (A1), that is, assumption (A3) follows from (A1) by taking $\mathbf{u}' = \pi(\mathbf{u})$. Then by (A.42), for $\varepsilon' > \delta$, we can estimate the corresponding sample size required to solve the reduced optimization problem, obtained by replacing $\mathscr{U}$ with $\mathscr{U}_\tau$, as

$$N \geqslant \frac{2\sigma^2}{(\varepsilon' - \delta)^2}\left[n(\ln D - \ln \tau) + \ln\left(\mathcal{O}(1)/\alpha\right)\right]. \tag{A.48}$$

Suppose further that there exists a function $\kappa : \mathscr{X} \to \mathbb{R}_+$ and $\varrho > 0$ such that

$$|H(\mathbf{X}; \mathbf{u}') - H(\mathbf{X}; \mathbf{u})| \leqslant \kappa(\mathbf{X})\|\mathbf{u}' - \mathbf{u}\|^{\varrho} \tag{A.49}$$

holds for all $\mathbf{u}', \mathbf{u} \in \mathscr{U}$ and all $\mathbf{X} \in \mathscr{X}$. It follows by (A.49) that

$$|\widehat{\ell}(\mathbf{u}') - \widehat{\ell}(\mathbf{u})| \leqslant N^{-1}\sum_{j=1}^{N} |H(\mathbf{X}_j; \mathbf{u}') - H(\mathbf{X}_j; \mathbf{u})| \leqslant \widehat{\kappa}\,\|\mathbf{u}' - \mathbf{u}\|^{\varrho}, \tag{A.50}$$

where $\widehat{\kappa} = N^{-1}\sum_{j=1}^{N} \kappa(\mathbf{X}_j)$.

Let us further assume the following:

(A4) *The moment generating function* $M_\kappa(t) = \mathbb{E}\left[e^{t\kappa(\mathbf{X})}\right]$ *of* $\kappa(\mathbf{X})$ *is finite-valued for all* t *in a neighborhood of* 0.

It follows then that the expectation $L = \mathbb{E}[\kappa(\mathbf{X})]$ is finite, and moreover, by Cramér's large deviations theorem that for any $L' > L$ there exists a positive constant $\beta = \beta(L')$ such that

$$\mathbb{P}(\widehat{\kappa} > L') \leqslant \mathrm{e}^{-N\beta} . \tag{A.51}$$

Let $\widehat{\mathbf{u}}$ be a δ-optimal solution of the stochastic counterpart problem and let $\tilde{\mathbf{u}} \in \mathscr{U}_\tau$ be a point such that $\|\widehat{\mathbf{u}} - \tilde{\mathbf{u}}\| \leqslant \tau$. Let us take $N \geqslant \beta^{-1}\ln(2/\alpha)$, so that by (A.51) we have

$$\mathbb{P}(\widehat{\kappa} > L') \leqslant \alpha/2 . \tag{A.52}$$

Then with probability at least $1 - \alpha/2$, the point $\tilde{\mathbf{u}}$ is a $(\delta + L'\tau^\varrho)$-optimal solution of the reduced stochastic counterpart problem. Setting

$$\tau = [(\varepsilon - \delta)/(2L')]^{1/\varrho} ,$$

we find that with probability at least $1 - \alpha/2$, the point $\tilde{\mathbf{u}}$ is an ε'-optimal solution of the reduced stochastic counterpart problem with $\varepsilon' = (\varepsilon + \delta)/2$. Moreover, by taking a sample size satisfying (A.48), we find that $\tilde{\mathbf{u}}$ is an ε'-optimal solution of the reduced expected-value problem with probability at least $1 - \alpha/2$. It follows that $\widehat{\mathbf{u}}$ is an ε''-optimal solution of the stochastic counterpart problem (A.31) with probability at least $1 - \alpha$ and $\varepsilon'' = \varepsilon' + L\tau^\varrho \leqslant \varepsilon$. We obtain the following estimate

$$N \geqslant \frac{4\sigma^2}{(\varepsilon - \delta)^2}\left[n\left(\ln D + \varrho^{-1}\ln\frac{2L'}{\varepsilon - \delta}\right) + \ln\left(\frac{\mathcal{O}(1)}{\alpha}\right)\right] \vee \left[\beta^{-1}\ln(2/\alpha)\right] \tag{A.53}$$

for the sample size, where $\vee$ denotes the maximum.

The above result is quite general and does not involve the convexity assumption. The estimate (A.53) of the sample size contains various constants and is too conservative for practical applications. However, it can be used as an estimate of the complexity of two-stage stochastic programming problems. In typical applications (e.g., in the convex case) the constant $\varrho = 1$, in which case condition (A.49) means that $H(\mathbf{X}; \cdot)$ is Lipschitz continuous on $\mathscr{U}$ with constant $\kappa(\mathbf{X})$. Note that there are also some applications where ϱ could be less than 1. We obtain the following basic result.

Theorem A.8.1 *Suppose that assumptions* (A3) *and* (A4) *hold and* $\mathscr{U}$ *has a finite diameter* D. *Then for* $\varepsilon > 0$, $0 \leqslant \delta < \varepsilon$ *and sample size* N *satisfying* (A.53), *we are guaranteed that any* δ*-optimal solution of the stochastic counterpart problem is an* ε*-optimal solution of the true problem with probability at least* $1 - \alpha$.

In particular, if we assume that $\varrho = 1$ and $\kappa(\mathbf{X}) = L$ for all $\mathbf{X} \in \mathscr{X}$, that is, $H(\mathbf{X}; \cdot)$ is Lipschitz continuous on $\mathscr{U}$ with constant L independent of $\mathbf{X} \in \mathscr{X}$, then we can take $\sigma = \mathcal{O}(1)DL$ and remove the term $\beta^{-1}\ln(2/\alpha)$ on the right-hand side of (A.53). Further, by taking $\delta = \varepsilon/2$ we find in that case the following estimate of the sample size (compare with estimate (A.46)):

$$N \geqslant \mathcal{O}(1)\left(\frac{DL}{\varepsilon}\right)^2\left[n\ln\left(\frac{DL}{\varepsilon}\right) + \ln\left(\frac{\mathcal{O}(1)}{\alpha}\right)\right] . \tag{A.54}$$

We can write the following simplified version of Theorem A.8.1.

Theorem A.8.2 *Suppose that $\mathscr{U}$ has a finite diameter D and condition* (A.49) *holds with $\varrho = 1$ and $\kappa(\mathbf{X}) = L$ for all $\mathbf{X} \in \mathscr{X}$. Then with sample size N satisfying* (A.54), *we are guaranteed that every $(\varepsilon/2)$-optimal solution of the stochastic counterpart problem is an ε-optimal solution of the true problem with probability at least $1 - \alpha$.*

The above estimates of the required sample size suggest complexity of order σ^2/ε^2 with respect to the desirable accuracy. This is in sharp contrast to deterministic (convex) optimization, where complexity usually is bounded in terms of $\ln(\varepsilon^{-1})$. In view of the above discussion, it should not be surprising that (even linear) two-stage stochastic programs usually cannot be solved with high accuracy. On the other hand, the estimates (A.53) and (A.54) depend *linearly* on the dimension n of the first-stage decision vector. They also depend linearly on $\ln(\alpha^{-1})$. This means that by increasing confidence, say, from 99% to 99.99%, we need to increase the sample size by a factor of $\ln 100 \approx 4.6$ at most. This also suggests that by using Monte Carlo sampling techniques, one can solve a two-stage stochastic program with reasonable accuracy, say with relative accuracy of 1% or 2%, in a reasonable time, provided that (a) its variability is not too large, (b) it has relatively complete recourse, and (c) the corresponding stochastic counterpart problem can be solved efficiently. And indeed, this was verified in numerical experiments with two-stage problems having a linear second-stage recourse. Of course, the estimate (A.53) of the sample size is far too conservative for the actual calculations. For practical applications, there are techniques that allow us to estimate the error of the feasible solution $\bar{\mathbf{u}}$ for a given sample size N; see, for example, [6].

The above estimates of the sample size are quite general. For convex problems, these bounds can be tightened in some cases. That is, suppose that the problem is convex, that is, the set $\mathscr{U}$ is convex and functions $H(\mathbf{X}; \cdot)$ are convex for all $\mathbf{X} \in \mathscr{X}$. Suppose further that $\kappa(\mathbf{X}) \equiv L$, the set $\mathscr{U}^0$, of optimal solutions of the true problem, is nonempty and bounded and for some $r \geqslant 1$, $c > 0$ and $a > 0$, the following growth condition holds:

$$\ell(\mathbf{u}) \geqslant \ell^* + c\,[\operatorname{dist}(\mathbf{u}, \mathscr{U}^0)]^r, \quad \forall\, \mathbf{u} \in \mathscr{U}^a\,, \tag{A.55}$$

where $a > 0$ and $\mathscr{U}^a = \{\mathbf{u} \in \mathscr{U} : \ell(\mathbf{u}) \leqslant \ell^* + a\}$ is the set of a-optimal solutions of the true problem. Then for any $\varepsilon \in (0, a)$ and $\delta \in [0, \varepsilon/2)$ we have the following estimate of the required sample size:

$$N \geqslant \left(\frac{\mathcal{O}(1)L}{c^{1/r}\varepsilon^{(r-1)/r}}\right)^2 \left[n \ln\left(\frac{\mathcal{O}(1)LD_a^*}{\varepsilon}\right) + \ln\left(\frac{1}{\alpha}\right)\right], \tag{A.56}$$

where D_a^* is the diameter of $\mathscr{U}^a$. Note that if $\mathscr{U}^0 = \{\mathbf{u}^*\}$ is a singleton, then it follows from (A.55) that $D_a^* \leqslant 2(a/c)^{1/r}$.

In particular, if $r = 1$ and $\mathscr{U}^0 = \{\mathbf{u}^*\}$ is a singleton, that is, the solution $\mathbf{u}^*$ is *sharp*, then D_a^* can be bounded by $4c^{-1}\varepsilon$ and hence we obtain the following estimate:

$$N \geqslant \mathcal{O}(1)c^{-2}L^2\left[n \ln\left(\mathcal{O}(1)c^{-1}L\right) + \ln\left(\alpha^{-1}\right)\right], \tag{A.57}$$

which does not depend on ε. That is, in that case, convergence to the exact optimal solution $\mathbf{u}^*$ happens with probability 1 in finite time.

For $r = 2$, condition (A.55) is called the *second-order* or *quadratic* growth condition. Under the quadratic growth condition, the first term on the right-hand side of (A.56) becomes of order $c^{-1}L^2\varepsilon^{-1}$.

PROBLEMS

A.1 Prove (A.8).

A.2 Let X and Y be Gaussian random vectors, with joint distribution given by

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathsf{N}\left(\underbrace{\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}}_{\mu}, \underbrace{\begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}}_{\Sigma} \right).$$

a) Defining $S = \Sigma_{12}\Sigma_{22}^{-1}$, show that

$$\begin{pmatrix} I & -S \\ 0 & I \end{pmatrix} \Sigma \begin{pmatrix} I & 0 \\ -S^T & I \end{pmatrix} = \begin{pmatrix} \Sigma_{11} - S\Sigma_{21} & 0 \\ 0 & \Sigma_{22} \end{pmatrix}.$$

b) Using the above result, show that for any vectors u and v

$$(u^T\, v^T)\Sigma^{-1} \begin{pmatrix} u \\ v \end{pmatrix} = (u^T - v^T S^T)\widetilde{\Sigma}^{-1}(u - Sv) + v^T\Sigma_{22}^{-1}v\,,$$

where $\widetilde{\Sigma} = (\Sigma_{11} - S\Sigma_{21})$.

c) The joint pdf of X and Y is given by

$$f(x, y) = c_1 \exp\left[-\frac{1}{2}(x^T - \mu_1^T \quad y^T - \mu_2^T)\,\Sigma^{-1} \begin{pmatrix} x - \mu_1 \\ y - \mu_2 \end{pmatrix}\right]$$

for some constant c_1. Using b), show that the conditional pdf $f(x \mid y)$ is of the form

$$f(x \mid y) = c_2(y) \exp\left[-\frac{1}{2}(x^T - \tilde{\mu}^T)\,\widetilde{\Sigma}^{-1}\,(x - \tilde{\mu})\right],$$

with $\tilde{\mu} = \mu_1 + S(y - \mu_2)$, and where $c_2(y)$ is some function of y (need not be specified). This proves that

$$(X \mid Y = y) \sim \mathsf{N}\left(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y - \mu_2),\ \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{12}^T\right).$$

Further Reading

More details on exponential families and their role in statistics may be found in [1]. An accessible account of hidden Markov models is [3].

The estimate (A.42) of the sample size, for finite feasible set $\mathscr{U}$, was obtained in [4]. For a general discussion of such estimates and extensions to the general case, see [6]. For a discussion of the complexity of *multistage* stochastic programming problems, see, for example, [8]. Finite time convergence in cases of sharp optimal solutions is discussed in [7].

REFERENCES

1. G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, 2nd edition, 2001.
2. S. X. Chen and J. S. Liu. Statistical applications of the Poisson-binomial and conditional Bernoulli distributions. *Statistica Sinica*, 7:875–892, 1997.
3. R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classifi cation*. John Wiley & Sons, New York, 2001.
4. A. J. Kleywegt, A. Shapiro, and T. Homem de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12:479–502, 2001.
5. R. Y. Rubinstein and A. Shapiro. *Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization via the Score Function Method*. John Wiley & Sons, New York, 1993.
6. A. Shapiro. Monte Carlo sampling methods. In A. Ruszczyński and A. Shapiro, editors, *Handbook in Operations Research and Management Science*, volume 10. Elsevier, Amsterdam, 2003.
7. A. Shapiro and T. Homem de Mello. On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs. *SIAM Journal on Optimization*, 11(1):70–86, 2001.
8. A. Shapiro and A. Nemirovski. On complexity of stochastic programming problems. In V. Jeyakumar and A.M. Rubinov, editors, *Continuous Optimization: Current Trends and Application*. Springer-Verlag, New York, 2005.

ABBREVIATIONS AND ACRONYMS

cdf	cumulative distribution function
CE	cross-entropy
CMC	crude Monte Carlo
CNF	conjunctive normal form
DEDS	discrete-event dynamical system
DESS	discrete-event statical system
DES	discrete-event simulation
DNF	disjunctive normal form
ECM	exponential change of measure
FPAUS	fully polynomial almost uniform sampler
FPRAS	fully polynomial randomized approximation scheme
HMM	hidden Markov model
iid	independent and identically distributed
ITLR	inverse-transform likelihood ratio
KKT	Karush-Kuhn-Tucker
max-cut	maximal cut
MCMC	Markov chain Monte Carlo
MinxEnt	minimum cross-entropy

pdf	probability density function (both discrete and continuous)
PERT	program evaluation and review technique
PME	parametric MinxEnt
RSAT	random SAT
SAT	satisfiability (problem)
SF	score function
SIS	sequential importance sampling
SLR	standard likelihood ratio
TLR	transform likelihood ratio
TSP	traveling salesman problem
VM	variance minimization

LIST OF SYMBOLS

$\gg$	much greater than
$\propto$	proportional to
$\sim$	is distributed according to
$\approx$	approximately
∇	∇f is the gradient of f
∇^2	$\nabla^2 f$ is the Hessian of f
$\mathbb{E}$	expectation
$\mathbb{N}$	set of natural numbers $\{0, 1, \ldots\}$
$\mathbb{P}$	probability measure
$\mathbb{R}$	the real line = one-dimensional Euclidean space
$\mathbb{R}^n$	n-dimensional Euclidean space
$\mathcal{D}$	Kullback-Leibler CE
$\mathcal{H}$	Shannon entropy
$\mathcal{M}$	mutual information
$\mathcal{S}$	score function

Ber	Bernoulli distribution
Beta	beta distribution
Bin	binomial distribution
Exp	exponential distribution
G	geometric distribution
Gamma	gamma distribution
N	normal or Gaussian distribution
Pareto	Pareto distribution
Poi	Poisson distribution
U	uniform distribution
Weib	Weibull distribution

α	smoothing parameter or acceptance probability
γ	level parameter
ζ	cumulant function (log of moment generating function)
ϱ	rarity parameter

$D(\mathbf{v})$	objective function for CE minimization
f	probability density (discrete or continuous)
g	importance sampling density
I_A	indicator function of event A
ln	(natural) logarithm
N	sample size
$\mathcal{O}$	Big-O order symbol
S	performance function
$S_{(i)}$	i-th order statistic
$\mathbf{u}$	nominal reference parameter (vector)
$\mathbf{v}$	reference parameter (vector)
$\widehat{\mathbf{v}}$	estimated reference parameter
$\mathbf{v}^*$	CE optimal reference parameter
${}_*\mathbf{v}$	VM optimal reference parameter
$V(\mathbf{v})$	objective function for VM minimization
W	likelihood ratio
$\mathbf{x}, \mathbf{y}$	vectors
$\mathbf{X}, \mathbf{Y}$	random vectors
$\mathscr{X}, \mathscr{Y}$	sets

This Page Intentionally Left Blank

INDEX

WILEY SERIES IN PROBABILITY AND STATISTICS

Established by WALTER A. SHEWHART and SAMUEL S. WILKS

Editors: *David J. Balding, Noel A. C. Cressie, Nicholas I. Fisher, Iain M. Johnstone, J. B. Kadane, Geert Molenberghs, David W. Scott, Adrian F. M. Smith, Sanford Weisberg*
Editors Emeriti: *Vic Barnett, J. Stuart Hunter, David G. Kendall, Jozef L. Teugels*

A complete list of the titles in this series appears at the end of this volume.

WILEY SERIES IN PROBABILITY AND STATISTICS

ESTABLISHED BY WALTER A. SHEWHART AND SAMUEL S. WILKS

The ***Wiley Series in Probability and Statistics*** is well established and authoritative. It covers many topics of current research interest in both pure and applied statistics and probability theory. Written by leading statisticians and institutions, the titles span both state-of-the-art developments in the field and classical methods.

Reflecting the wide range of current research in statistics, the series encompasses applied, methodological and theoretical statistics, ranging from applications and new techniques made possible by advances in computerized practice to rigorous treatment of theoretical approaches.

This series provides essential and invaluable reading for all statisticians, whether in academia, industry, government, or research.

† ABRAHAM and LEDOLTER · Statistical Methods for Forecasting
AGRESTI · Analysis of Ordinal Categorical Data
AGRESTI · An Introduction to Categorical Data Analysis, *Second Edition*
AGRESTI · Categorical Data Analysis, *Second Edition*
ALTMAN, GILL, and McDONALD · Numerical Issues in Statistical Computing for the Social Scientist
AMARATUNGA and CABRERA · Exploration and Analysis of DNA Microarray and Protein Array Data
ANDĚL · Mathematics of Chance
ANDERSON · An Introduction to Multivariate Statistical Analysis, *Third Edition*
* ANDERSON · The Statistical Analysis of Time Series
ANDERSON, AUQUIER, HAUCK, OAKES, VANDAELE, and WEISBERG · Statistical Methods for Comparative Studies
ANDERSON and LOYNES · The Teaching of Practical Statistics
ARMITAGE and DAVID (editors) · Advances in Biometry
ARNOLD, BALAKRISHNAN, and NAGARAJA · Records
* ARTHANARI and DODGE · Mathematical Programming in Statistics
* BAILEY · The Elements of Stochastic Processes with Applications to the Natural Sciences
BALAKRISHNAN and KOUTRAS · Runs and Scans with Applications
BALAKRISHNAN and NG · Precedence-Type Tests and Applications
BARNETT · Comparative Statistical Inference, *Third Edition*
BARNETT · Environmental Statistics
BARNETT and LEWIS · Outliers in Statistical Data, *Third Edition*
BARTOSZYNSKI and NIEWIADOMSKA-BUGAJ · Probability and Statistical Inference
BASILEVSKY · Statistical Factor Analysis and Related Methods: Theory and Applications
BASU and RIGDON · Statistical Methods for the Reliability of Repairable Systems
BATES and WATTS · Nonlinear Regression Analysis and Its Applications

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

BECHHOFER, SANTNER, and GOLDSMAN · Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons
BELSLEY · Conditioning Diagnostics: Collinearity and Weak Data in Regression
† BELSLEY, KUH, and WELSCH · Regression Diagnostics: Identifying Influential Data and Sources of Collinearity
BENDAT and PIERSOL · Random Data: Analysis and Measurement Procedures, *Third Edition*
BERRY, CHALONER, and GEWEKE · Bayesian Analysis in Statistics and Econometrics: Essays in Honor of Arnold Zellner
BERNARDO and SMITH · Bayesian Theory
BHAT and MILLER · Elements of Applied Stochastic Processes, *Third Edition*
BHATTACHARYA and WAYMIRE · Stochastic Processes with Applications
BILLINGSLEY · Convergence of Probability Measures, *Second Edition*
BILLINGSLEY · Probability and Measure, *Third Edition*
BIRKES and DODGE · Alternative Methods of Regression
BISWAS, DATTA, FINE, and SEGAL · Statistical Advances in the Biomedical Sciences: Clinical Trials, Epidemiology, Survival Analysis, and Bioinformatics
BLISCHKE AND MURTHY (editors) · Case Studies in Reliability and Maintenance
BLISCHKE AND MURTHY · Reliability: Modeling, Prediction, and Optimization
BLOOMFIELD · Fourier Analysis of Time Series: An Introduction, *Second Edition*
BOLLEN · Structural Equations with Latent Variables
BOLLEN and CURRAN · Latent Curve Models: A Structural Equation Perspective
BOROVKOV · Ergodicity and Stability of Stochastic Processes
BOULEAU · Numerical Methods for Stochastic Processes
BOX · Bayesian Inference in Statistical Analysis
BOX · R. A. Fisher, the Life of a Scientist
BOX and DRAPER · Response Surfaces, Mixtures, and Ridge Analyses, *Second Edition*
* BOX and DRAPER · Evolutionary Operation: A Statistical Method for Process Improvement
BOX and FRIENDS · Improving Almost Anything, *Revised Edition*
BOX, HUNTER, and HUNTER · Statistics for Experimenters: Design, Innovation, and Discovery, *Second Editon*
BOX and LUCEÑO · Statistical Control by Monitoring and Feedback Adjustment
BRANDIMARTE · Numerical Methods in Finance: A MATLAB-Based Introduction
† BROWN and HOLLANDER · Statistics: A Biomedical Introduction
BRUNNER, DOMHOF, and LANGER · Nonparametric Analysis of Longitudinal Data in Factorial Experiments
BUCKLEW · Large Deviation Techniques in Decision, Simulation, and Estimation
CAIROLI and DALANG · Sequential Stochastic Optimization
CASTILLO, HADI, BALAKRISHNAN, and SARABIA · Extreme Value and Related Models with Applications in Engineering and Science
CHAN · Time Series: Applications to Finance
CHARALAMBIDES · Combinatorial Methods in Discrete Distributions
CHATTERJEE and HADI · Regression Analysis by Example, *Fourth Edition*
CHATTERJEE and HADI · Sensitivity Analysis in Linear Regression
CHERNICK · Bootstrap Methods: A Guide for Practitioners and Researchers, *Second Edition*
CHERNICK and FRIIS · Introductory Biostatistics for the Health Sciences
CHILÈS and DELFINER · Geostatistics: Modeling Spatial Uncertainty
CHOW and LIU · Design and Analysis of Clinical Trials: Concepts and Methodologies, *Second Edition*
CLARKE and DISNEY · Probability and Random Processes: A First Course with Applications, *Second Edition*
* COCHRAN and COX · Experimental Designs, *Second Edition*

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

CONGDON · Applied Bayesian Modelling
CONGDON · Bayesian Models for Categorical Data
CONGDON · Bayesian Statistical Modelling
CONOVER · Practical Nonparametric Statistics, *Third Edition*
COOK · Regression Graphics
COOK and WEISBERG · Applied Regression Including Computing and Graphics
COOK and WEISBERG · An Introduction to Regression Graphics
CORNELL · Experiments with Mixtures, Designs, Models, and the Analysis of Mixture Data, *Third Edition*
COVER and THOMAS · Elements of Information Theory
COX · A Handbook of Introductory Statistical Methods
* COX · Planning of Experiments
CRESSIE · Statistics for Spatial Data, *Revised Edition*
CSÖRGŐ and HORVÁTH · Limit Theorems in Change Point Analysis
DANIEL · Applications of Statistics to Industrial Experimentation
DANIEL · Biostatistics: A Foundation for Analysis in the Health Sciences, *Eighth Edition*
* DANIEL · Fitting Equations to Data: Computer Analysis of Multifactor Data, *Second Edition*
DASU and JOHNSON · Exploratory Data Mining and Data Cleaning
DAVID and NAGARAJA · Order Statistics, *Third Edition*
* DEGROOT, FIENBERG, and KADANE · Statistics and the Law
DEL CASTILLO · Statistical Process Adjustment for Quality Control
DEMARIS · Regression with Social Data: Modeling Continuous and Limited Response Variables
DEMIDENKO · Mixed Models: Theory and Applications
DENISON, HOLMES, MALLICK and SMITH · Bayesian Methods for Nonlinear Classification and Regression
DETTE and STUDDEN · The Theory of Canonical Moments with Applications in Statistics, Probability, and Analysis
DEY and MUKERJEE · Fractional Factorial Plans
DILLON and GOLDSTEIN · Multivariate Analysis: Methods and Applications
DODGE · Alternative Methods of Regression
* DODGE and ROMIG · Sampling Inspection Tables, *Second Edition*
* DOOB · Stochastic Processes
DOWDY, WEARDEN, and CHILKO · Statistics for Research, *Third Edition*
DRAPER and SMITH · Applied Regression Analysis, *Third Edition*
DRYDEN and MARDIA · Statistical Shape Analysis
DUDEWICZ and MISHRA · Modern Mathematical Statistics
DUNN and CLARK · Basic Statistics: A Primer for the Biomedical Sciences, *Third Edition*
DUPUIS and ELLIS · A Weak Convergence Approach to the Theory of Large Deviations
EDLER and KITSOS · Recent Advances in Quantitative Methods in Cancer and Human Health Risk Assessment
* ELANDT-JOHNSON and JOHNSON · Survival Models and Data Analysis
ENDERS · Applied Econometric Time Series
† ETHIER and KURTZ · Markov Processes: Characterization and Convergence
EVANS, HASTINGS, and PEACOCK · Statistical Distributions, *Third Edition*
FELLER · An Introduction to Probability Theory and Its Applications, Volume I, *Third Edition,* Revised; Volume II, *Second Edition*
FISHER and VAN BELLE · Biostatistics: A Methodology for the Health Sciences
FITZMAURICE, LAIRD, and WARE · Applied Longitudinal Analysis
* FLEISS · The Design and Analysis of Clinical Experiments
FLEISS · Statistical Methods for Rates and Proportions, *Third Edition*

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

† FLEMING and HARRINGTON · Counting Processes and Survival Analysis
FULLER · Introduction to Statistical Time Series, *Second Edition*
† FULLER · Measurement Error Models
GALLANT · Nonlinear Statistical Models
GEISSER · Modes of Parametric Statistical Inference
GELMAN and MENG · Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives
GEWEKE · Contemporary Bayesian Econometrics and Statistics
GHOSH, MUKHOPADHYAY, and SEN · Sequential Estimation
GIESBRECHT and GUMPERTZ · Planning, Construction, and Statistical Analysis of Comparative Experiments
GIFI · Nonlinear Multivariate Analysis
GIVENS and HOETING · Computational Statistics
GLASSERMAN and YAO · Monotone Structure in Discrete-Event Systems
GNANADESIKAN · Methods for Statistical Data Analysis of Multivariate Observations, *Second Edition*
GOLDSTEIN and LEWIS · Assessment: Problems, Development, and Statistical Issues
GREENWOOD and NIKULIN · A Guide to Chi-Squared Testing
GROSS and HARRIS · Fundamentals of Queueing Theory, *Third Edition*
* HAHN and SHAPIRO · Statistical Models in Engineering
HAHN and MEEKER · Statistical Intervals: A Guide for Practitioners
HALD · A History of Probability and Statistics and their Applications Before 1750
HALD · A History of Mathematical Statistics from 1750 to 1930
† HAMPEL · Robust Statistics: The Approach Based on Influence Functions
HANNAN and DEISTLER · The Statistical Theory of Linear Systems
HEIBERGER · Computation for the Analysis of Designed Experiments
HEDAYAT and SINHA · Design and Inference in Finite Population Sampling
HEDEKER and GIBBONS · Longitudinal Data Analysis
HELLER · MACSYMA for Statisticians
HINKELMANN and KEMPTHORNE · Design and Analysis of Experiments, Volume 1: Introduction to Experimental Design, *Second Edition*
HINKELMANN and KEMPTHORNE · Design and Analysis of Experiments, Volume 2: Advanced Experimental Design
HOAGLIN, MOSTELLER, and TUKEY · Exploratory Approach to Analysis of Variance
* HOAGLIN, MOSTELLER, and TUKEY · Exploring Data Tables, Trends and Shapes
* HOAGLIN, MOSTELLER, and TUKEY · Understanding Robust and Exploratory Data Analysis
HOCHBERG and TAMHANE · Multiple Comparison Procedures
HOCKING · Methods and Applications of Linear Models: Regression and the Analysis of Variance, *Second Edition*
HOEL · Introduction to Mathematical Statistics, *Fifth Edition*
HOGG and KLUGMAN · Loss Distributions
HOLLANDER and WOLFE · Nonparametric Statistical Methods, *Second Edition*
HOSMER and LEMESHOW · Applied Logistic Regression, *Second Edition*
HOSMER and LEMESHOW · Applied Survival Analysis: Regression Modeling of Time to Event Data
† HUBER · Robust Statistics
HUBERTY · Applied Discriminant Analysis
HUBERTY and OLEJNIK · Applied MANOVA and Discriminant Analysis, *Second Edition*
HUNT and KENNEDY · Financial Derivatives in Theory and Practice, *Revised Edition*

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

HURD and MIAMEE · Periodically Correlated Random Sequences: Spectral Theory and Practice
HUSKOVA, BERAN, and DUPAC · Collected Works of Jaroslav Hajek—with Commentary
HUZURBAZAR · Flowgraph Models for Multistate Time-to-Event Data
IMAN and CONOVER · A Modern Approach to Statistics
† JACKSON · A User's Guide to Principle Components
JOHN · Statistical Methods in Engineering and Quality Assurance
JOHNSON · Multivariate Statistical Simulation
JOHNSON and BALAKRISHNAN · Advances in the Theory and Practice of Statistics: A Volume in Honor of Samuel Kotz
JOHNSON and BHATTACHARYYA · Statistics: Principles and Methods, *Fifth Edition*
JOHNSON and KOTZ · Distributions in Statistics
JOHNSON and KOTZ (editors) · Leading Personalities in Statistical Sciences: From the Seventeenth Century to the Present
JOHNSON, KOTZ, and BALAKRISHNAN · Continuous Univariate Distributions, Volume 1, *Second Edition*
JOHNSON, KOTZ, and BALAKRISHNAN · Continuous Univariate Distributions, Volume 2, *Second Edition*
JOHNSON, KOTZ, and BALAKRISHNAN · Discrete Multivariate Distributions
JOHNSON, KEMP, and KOTZ · Univariate Discrete Distributions, *Third Edition*
JUDGE, GRIFFITHS, HILL, LÜTKEPOHL, and LEE · The Theory and Practice of Econometrics, *Second Edition*
JUREČKOVÁ and SEN · Robust Statistical Procedures: Aymptotics and Interrelations
JUREK and MASON · Operator-Limit Distributions in Probability Theory
KADANE · Bayesian Methods and Ethics in a Clinical Trial Design
KADANE AND SCHUM · A Probabilistic Analysis of the Sacco and Vanzetti Evidence
KALBFLEISCH and PRENTICE · The Statistical Analysis of Failure Time Data, *Second Edition*
KARIYA and KURATA · Generalized Least Squares
KASS and VOS · Geometrical Foundations of Asymptotic Inference
† KAUFMAN and ROUSSEEUW · Finding Groups in Data: An Introduction to Cluster Analysis
KEDEM and FOKIANOS · Regression Models for Time Series Analysis
KENDALL, BARDEN, CARNE, and LE · Shape and Shape Theory
KHURI · Advanced Calculus with Applications in Statistics, *Second Edition*
KHURI, MATHEW, and SINHA · Statistical Tests for Mixed Linear Models
KLEIBER and KOTZ · Statistical Size Distributions in Economics and Actuarial Sciences
KLUGMAN, PANJER, and WILLMOT · Loss Models: From Data to Decisions, *Second Edition*
KLUGMAN, PANJER, and WILLMOT · Solutions Manual to Accompany Loss Models: From Data to Decisions, *Second Edition*
KOTZ, BALAKRISHNAN, and JOHNSON · Continuous Multivariate Distributions, Volume 1, *Second Edition*
KOVALENKO, KUZNETZOV, and PEGG · Mathematical Theory of Reliability of Time-Dependent Systems with Practical Applications
KOWALSKI and TU · Modern Applied U-Statistics
KVAM and VIDAKOVIC · Nonparametric Statistics with Applications to Science and Engineering
LACHIN · Biostatistical Methods: The Assessment of Relative Risks
LAD · Operational Subjective Statistical Methods: A Mathematical, Philosophical, and Historical Introduction
LAMPERTI · Probability: A Survey of the Mathematical Theory, *Second Edition*

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

LANGE, RYAN, BILLARD, BRILLINGER, CONQUEST, and GREENHOUSE · Case Studies in Biometry
LARSON · Introduction to Probability Theory and Statistical Inference, *Third Edition*
LAWLESS · Statistical Models and Methods for Lifetime Data, *Second Edition*
LAWSON · Statistical Methods in Spatial Epidemiology
LE · Applied Categorical Data Analysis
LE · Applied Survival Analysis
LEE and WANG · Statistical Methods for Survival Data Analysis, *Third Edition*
LEPAGE and BILLARD · Exploring the Limits of Bootstrap
LEYLAND and GOLDSTEIN (editors) · Multilevel Modelling of Health Statistics
LIAO · Statistical Group Comparison
LINDVALL · Lectures on the Coupling Method
LIN · Introductory Stochastic Analysis for Finance and Insurance
LINHART and ZUCCHINI · Model Selection
LITTLE and RUBIN · Statistical Analysis with Missing Data, *Second Edition*
LLOYD · The Statistical Analysis of Categorical Data
LOWEN and TEICH · Fractal-Based Point Processes
MAGNUS and NEUDECKER · Matrix Differential Calculus with Applications in Statistics and Econometrics, *Revised Edition*
MALLER and ZHOU · Survival Analysis with Long Term Survivors
MALLOWS · Design, Data, and Analysis by Some Friends of Cuthbert Daniel
MANN, SCHAFER, and SINGPURWALLA · Methods for Statistical Analysis of Reliability and Life Data
MANTON, WOODBURY, and TOLLEY · Statistical Applications Using Fuzzy Sets
MARCHETTE · Random Graphs for Statistical Pattern Recognition
MARDIA and JUPP · Directional Statistics
MASON, GUNST, and HESS · Statistical Design and Analysis of Experiments with Applications to Engineering and Science, *Second Edition*
McCULLOCH and SEARLE · Generalized, Linear, and Mixed Models
McFADDEN · Management of Data in Clinical Trials, *Second Edition*
* McLACHLAN · Discriminant Analysis and Statistical Pattern Recognition
McLACHLAN, DO, and AMBROISE · Analyzing Microarray Gene Expression Data
McLACHLAN and KRISHNAN · The EM Algorithm and Extensions, *Second Edition*
McLACHLAN and PEEL · Finite Mixture Models
McNEIL · Epidemiological Research Methods
MEEKER and ESCOBAR · Statistical Methods for Reliability Data
MEERSCHAERT and SCHEFFLER · Limit Distributions for Sums of Independent Random Vectors: Heavy Tails in Theory and Practice
MICKEY, DUNN, and CLARK · Applied Statistics: Analysis of Variance and Regression, *Third Edition*
* MILLER · Survival Analysis, *Second Edition*
MONTGOMERY, PECK, and VINING · Introduction to Linear Regression Analysis, *Fourth Edition*
MORGENTHALER and TUKEY · Configural Polysampling: A Route to Practical Robustness
MUIRHEAD · Aspects of Multivariate Statistical Theory
MULLER and STOYAN · Comparison Methods for Stochastic Models and Risks
MURRAY · X-STAT 2.0 Statistical Experimentation, Design Data Analysis, and Nonlinear Optimization
MURTHY, XIE, and JIANG · Weibull Models
MYERS and MONTGOMERY · Response Surface Methodology: Process and Product Optimization Using Designed Experiments, *Second Edition*
MYERS, MONTGOMERY, and VINING · Generalized Linear Models. With Applications in Engineering and the Sciences

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

† NELSON · Accelerated Testing, Statistical Models, Test Plans, and Data Analyses
† NELSON · Applied Life Data Analysis
NEWMAN · Biostatistical Methods in Epidemiology
OCHI · Applied Probability and Stochastic Processes in Engineering and Physical Sciences
OKABE, BOOTS, SUGIHARA, and CHIU · Spatial Tesselations: Concepts and Applications of Voronoi Diagrams, *Second Edition*
OLIVER and SMITH · Influence Diagrams, Belief Nets and Decision Analysis
PALTA · Quantitative Methods in Population Health: Extensions of Ordinary Regressions
PANJER · Operational Risk: Modeling and Analytics
PANKRATZ · Forecasting with Dynamic Regression Models
PANKRATZ · Forecasting with Univariate Box-Jenkins Models: Concepts and Cases
* PARZEN · Modern Probability Theory and Its Applications
PEÑA, TIAO, and TSAY · A Course in Time Series Analysis
PIANTADOSI · Clinical Trials: A Methodologic Perspective
PORT · Theoretical Probability for Applications
POURAHMADI · Foundations of Time Series Analysis and Prediction Theory
POWELL · Approximate Dynamic Programming: Solving the Curses of Dimensionality
PRESS · Bayesian Statistics: Principles, Models, and Applications
PRESS · Subjective and Objective Bayesian Statistics, *Second Edition*
PRESS and TANUR · The Subjectivity of Scientists and the Bayesian Approach
PUKELSHEIM · Optimal Experimental Design
PURI, VILAPLANA, and WERTZ · New Perspectives in Theoretical and Applied Statistics
† PUTERMAN · Markov Decision Processes: Discrete Stochastic Dynamic Programming
QIU · Image Processing and Jump Regression Analysis
* RAO · Linear Statistical Inference and Its Applications, *Second Edition*
RAUSAND and HØYLAND · System Reliability Theory: Models, Statistical Methods, and Applications, *Second Edition*
RENCHER · Linear Models in Statistics
RENCHER · Methods of Multivariate Analysis, *Second Edition*
RENCHER · Multivariate Statistical Inference with Applications
* RIPLEY · Spatial Statistics
* RIPLEY · Stochastic Simulation
ROBINSON · Practical Strategies for Experimenting
ROHATGI and SALEH · An Introduction to Probability and Statistics, *Second Edition*
ROLSKI, SCHMIDLI, SCHMIDT, and TEUGELS · Stochastic Processes for Insurance and Finance
ROSENBERGER and LACHIN · Randomization in Clinical Trials: Theory and Practice
ROSS · Introduction to Probability and Statistics for Engineers and Scientists
ROSSI, ALLENBY, and McCULLOCH · Bayesian Statistics and Marketing
† ROUSSEEUW and LEROY · Robust Regression and Outlier Detection
* RUBIN · Multiple Imputation for Nonresponse in Surveys
RUBINSTEIN and KROESE · Simulation and the Monte Carlo Method, *Second Edition*
RUBINSTEIN and MELAMED · Modern Simulation and Modeling
RYAN · Modern Engineering Statistics
RYAN · Modern Experimental Design
RYAN · Modern Regression Methods
RYAN · Statistical Methods for Quality Improvement, *Second Edition*
SALEH · Theory of Preliminary Test and Stein-Type Estimation with Applications
* SCHEFFE · The Analysis of Variance
SCHIMEK · Smoothing and Regression: Approaches, Computation, and Application
SCHOTT · Matrix Analysis for Statistics, *Second Edition*
SCHOUTENS · Levy Processes in Finance: Pricing Financial Derivatives

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

SCHUSS · Theory and Applications of Stochastic Differential Equations
SCOTT · Multivariate Density Estimation: Theory, Practice, and Visualization
† SEARLE · Linear Models for Unbalanced Data
† SEARLE · Matrix Algebra Useful for Statistics
† SEARLE, CASELLA, and McCULLOCH · Variance Components
SEARLE and WILLETT · Matrix Algebra for Applied Economics
SEBER · A Matrix Handbook For Statisticians
† SEBER · Multivariate Observations
SEBER and LEE · Linear Regression Analysis, *Second Edition*
† SEBER and WILD · Nonlinear Regression
SENNOTT · Stochastic Dynamic Programming and the Control of Queueing Systems
* SERFLING · Approximation Theorems of Mathematical Statistics
SHAFER and VOVK · Probability and Finance: It's Only a Game!
SILVAPULLE and SEN · Constrained Statistical Inference: Inequality, Order, and Shape Restrictions
SMALL and McLEISH · Hilbert Space Methods in Probability and Statistical Inference
SRIVASTAVA · Methods of Multivariate Statistics
STAPLETON · Linear Statistical Models
STAPLETON · Models for Probability and Statistical Inference: Theory and Applications
STAUDTE and SHEATHER · Robust Estimation and Testing
STOYAN, KENDALL, and MECKE · Stochastic Geometry and Its Applications, *Second Edition*
STOYAN and STOYAN · Fractals, Random Shapes and Point Fields: Methods of Geometrical Statistics
STREET and BURGESS · The Construction of Optimal Stated Choice Experiments: Theory and Methods
STYAN · The Collected Papers of T. W. Anderson: 1943–1985
SUTTON, ABRAMS, JONES, SHELDON, and SONG · Methods for Meta-Analysis in Medical Research
TAKEZAWA · Introduction to Nonparametric Regression
TANAKA · Time Series Analysis: Nonstationary and Noninvertible Distribution Theory
THOMPSON · Empirical Model Building
THOMPSON · Sampling, *Second Edition*
THOMPSON · Simulation: A Modeler's Approach
THOMPSON and SEBER · Adaptive Sampling
THOMPSON, WILLIAMS, and FINDLAY · Models for Investors in Real World Markets
TIAO, BISGAARD, HILL, PEÑA, and STIGLER (editors) · Box on Quality and Discovery: with Design, Control, and Robustness
TIERNEY · LISP-STAT: An Object-Oriented Environment for Statistical Computing and Dynamic Graphics
TSAY · Analysis of Financial Time Series, *Second Edition*
UPTON and FINGLETON · Spatial Data Analysis by Example, Volume II: Categorical and Directional Data
VAN BELLE · Statistical Rules of Thumb
VAN BELLE, FISHER, HEAGERTY, and LUMLEY · Biostatistics: A Methodology for the Health Sciences, *Second Edition*
VESTRUP · The Theory of Measures and Integration
VIDAKOVIC · Statistical Modeling by Wavelets
VINOD and REAGLE · Preparing for the Worst: Incorporating Downside Risk in Stock Market Investments
WALLER and GOTWAY · Applied Spatial Statistics for Public Health Data
WEERAHANDI · Generalized Inference in Repeated Measures: Exact Methods in MANOVA and Mixed Models
WEISBERG · Applied Linear Regression, *Third Edition*

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

WELSH · Aspects of Statistical Inference
WESTFALL and YOUNG · Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment
WHITTAKER · Graphical Models in Applied Multivariate Statistics
WINKER · Optimization Heuristics in Economics: Applications of Threshold Accepting
WONNACOTT and WONNACOTT · Econometrics, *Second Edition*
WOODING · Planning Pharmaceutical Clinical Trials: Basic Statistical Principles
WOODWORTH · Biostatistics: A Bayesian Introduction
WOOLSON and CLARKE · Statistical Methods for the Analysis of Biomedical Data, *Second Edition*
WU and HAMADA · Experiments: Planning, Analysis, and Parameter Design Optimization
WU and ZHANG · Nonparametric Regression Methods for Longitudinal Data Analysis
YANG · The Construction Theory of Denumerable Markov Processes
YOUNG, VALERO-MORA, and FRIENDLY · Visual Statistics: Seeing Data with Dynamic Interactive Graphics
ZELTERMAN · Discrete Distributions—Applications in the Health Sciences
* ZELLNER · An Introduction to Bayesian Inference in Econometrics
ZHOU, OBUCHOWSKI, and McCLISH · Statistical Methods in Diagnostic Medicine

*Now available in a lower priced paperback edition in the Wiley Classics Library.
†Now available in a lower priced paperback edition in the Wiley–Interscience Paperback Series.

This Page Intentionally Left Blank